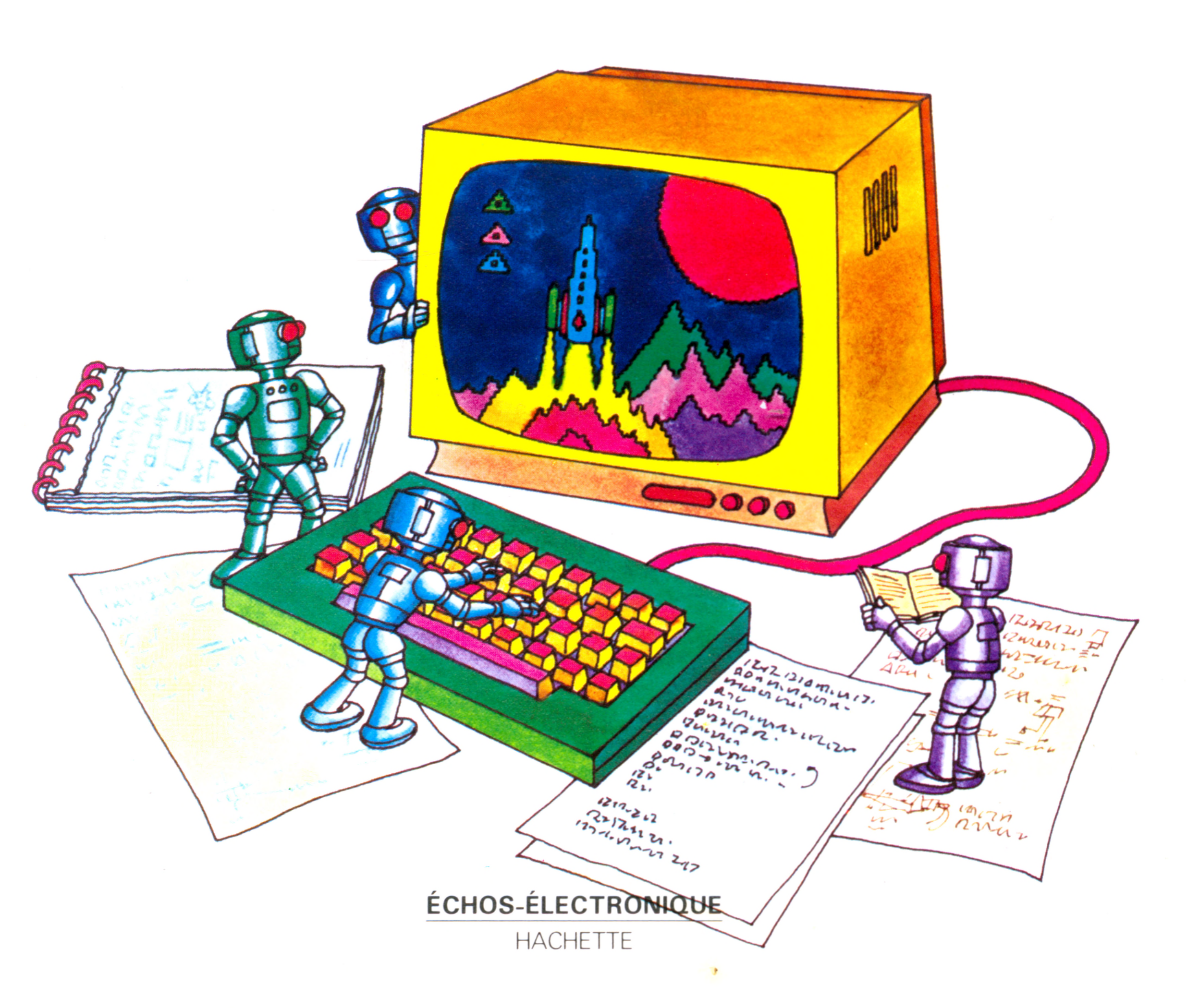

GUIDE PRATIQUE DU
BASIC
Initiation à la programmation
Hachette Jeunesse

GUIDE PRATIQUE DU
BASIC

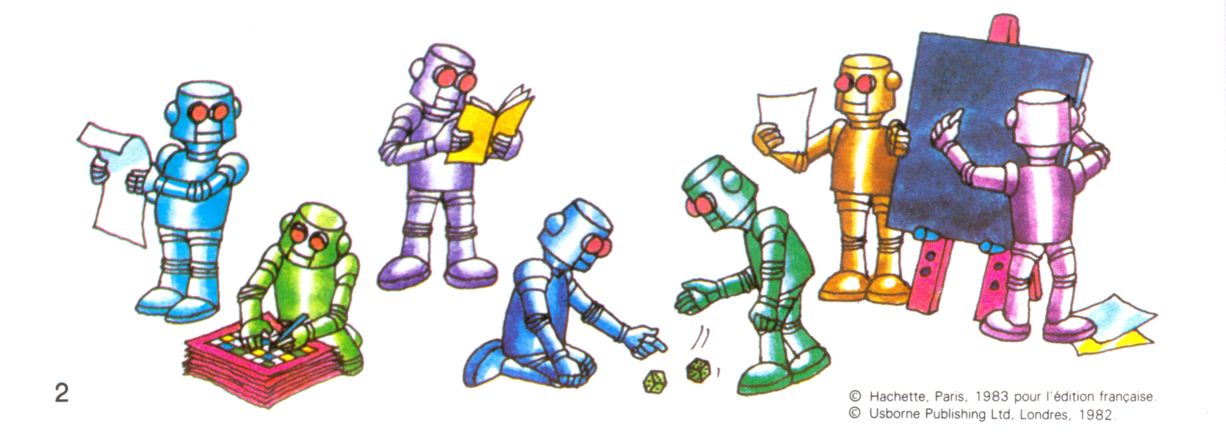
Brian Reffin Smith

Conception graphique : Kim Blundell. Illustrations de G. Round et M. Newton. Édition : Lisa Watts pour l'édition anglaise; Patrick Baradeau pour l'édition française. Révision : Jean-Noël Von der Weid. Assistance technique et Conseil : European Media Business, 9, place des Ternes, 75017 Paris.



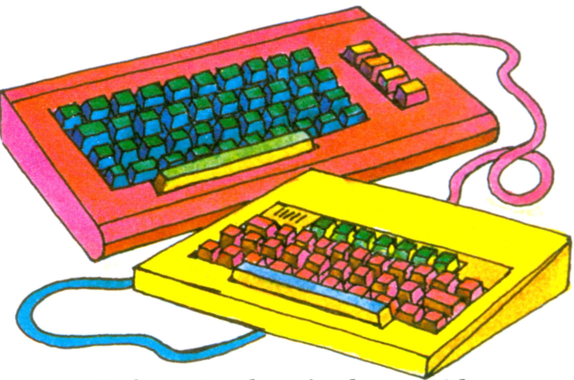
ÉCHOS-ÉLECTRONIQUE
HACHETTE

Sommaire

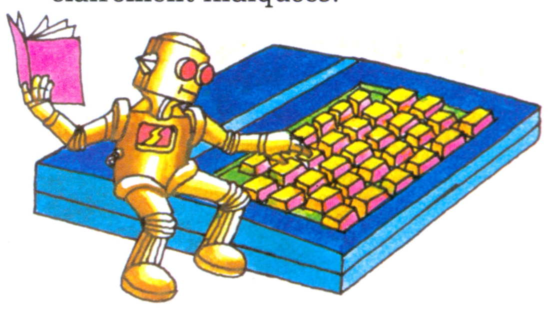


Avant-propos

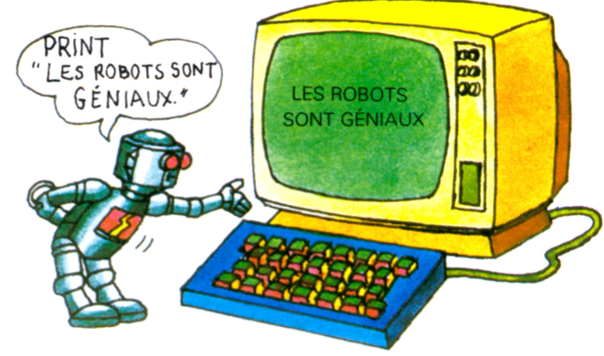
Ce guide permettra aux débutants d'apprendre à programmer des ordinateurs en BASIC. Le BASIC est le langage de la plupart des ordinateurs domestiques; c'est une façon de donner à la machine des instructions qu'elle comprend.



Point n'est besoin de posséder un ordinateur pour utiliser ce livre, même s'il est plus facile de comprendre les programmes en les essayant. Des différences de fabrication font que de légères variations interviennent dans le BASIC utilisé par tel ou tel appareil. Les instructions proposées ici sont acceptées par la plupart des micro-ordinateurs, et les rares exceptions sont clairement indiquées.



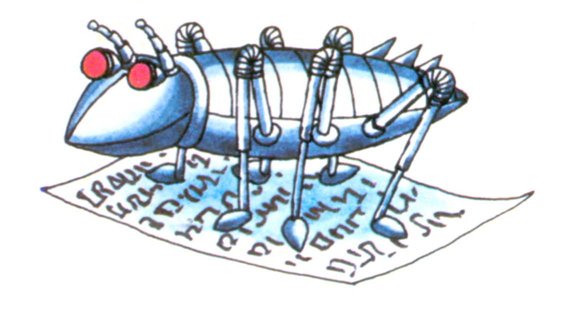
Pour commencer, vous trouverez une introduction générale à la programmation. Puis, au fil des pages, les différents mots-clés du BASIC et quelques programmes types permettant de les mettre en œuvre.



Pour vous entraîner à écrire des programmes, ce livre vous propose des casse-tête à résoudre, des suggestions pour créer ou modifier des programmes (les solutions sont données pages 44 et 45).

À la fin du livre (pages 46-47), un lexique des mots propres au BASIC et aux ordinateurs, des conseils pour vous aider à programmer, la liste des « bugs » (erreurs qui empêchent le programme de fonctionner) les plus fréquents et des trucs pour les reconnaître finiront de vous armer pour la programmation.

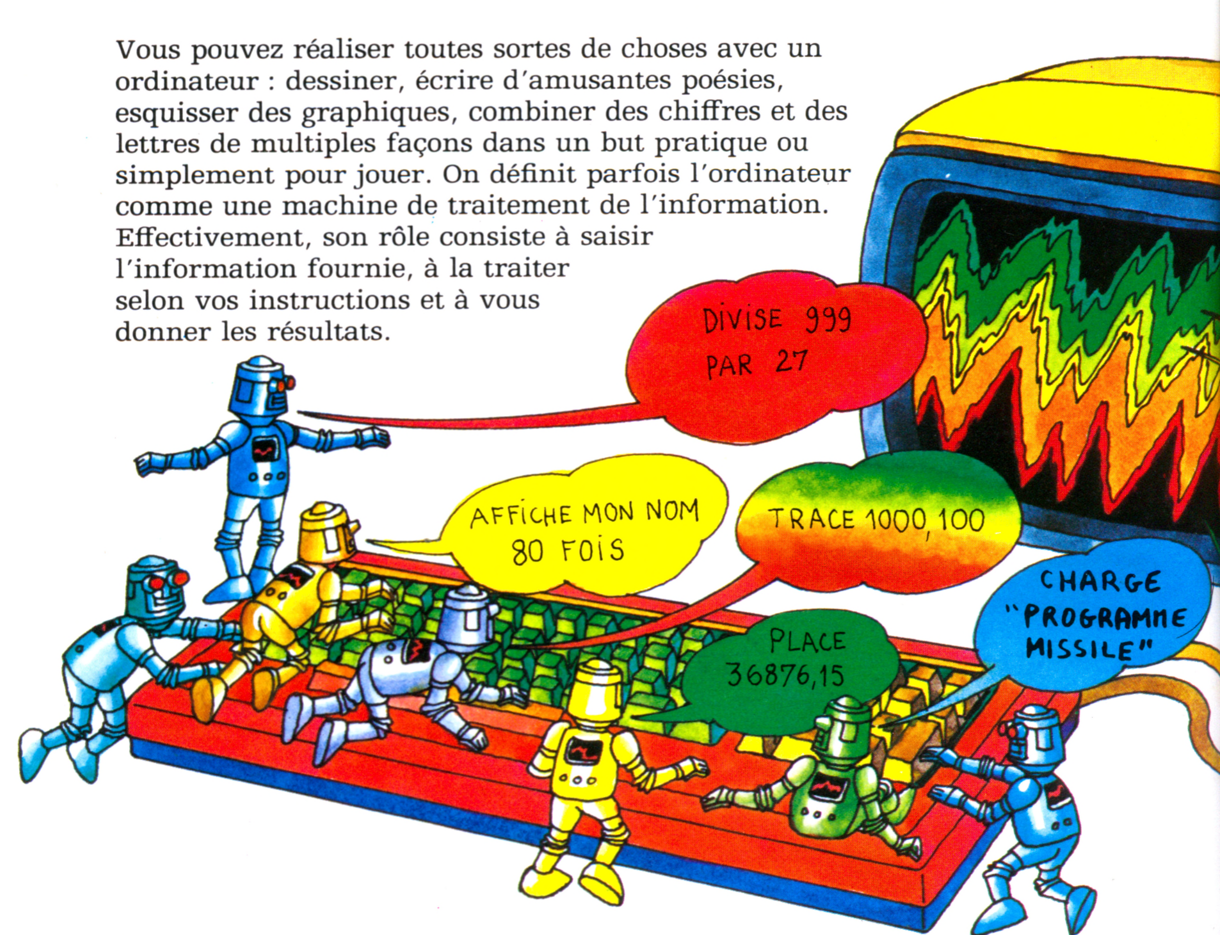
Si vous avez un micro, essayez les



programmes de ce livre, et relevez les instructions BASIC du manuel fourni par son fabricant; il se peut que certaines procédures indiquées ici ne soient pas nécessaires. La meilleure façon d'apprendre le BASIC? essayer de nombreux programmes proposés par les livres et les revues spécialisés, et parvenir à les transformer. Si vous suivez ces conseils, vous vous retrouverez rapidement en train d'écrire vos propres programmes !

Comment fonctionne un ordinateur

Vous pouvez réaliser toutes sortes de choses avec un ordinateur : dessiner, écrire d'amusantes poésies, esquisser des graphiques, combiner des chiffres et des lettres de multiples façons dans un but pratique ou simplement pour jouer. On définit parfois l'ordinateur comme une machine de traitement de l'information. Effectivement, son rôle consiste à saisir l'information fournie, à la traiter selon vos instructions et à vous donner les résultats.



L'ordinateur est un outil. Pour qu'il travaille, il faut lui donner des instructions extrêmement précises. La liste des instructions s'appelle un *programme*, les informations que vous lui fournissez, des *données*. Le programme doit être écrit dans un langage que l'ordinateur comprend, ainsi le BASIC, et en respecter toutes les règles.

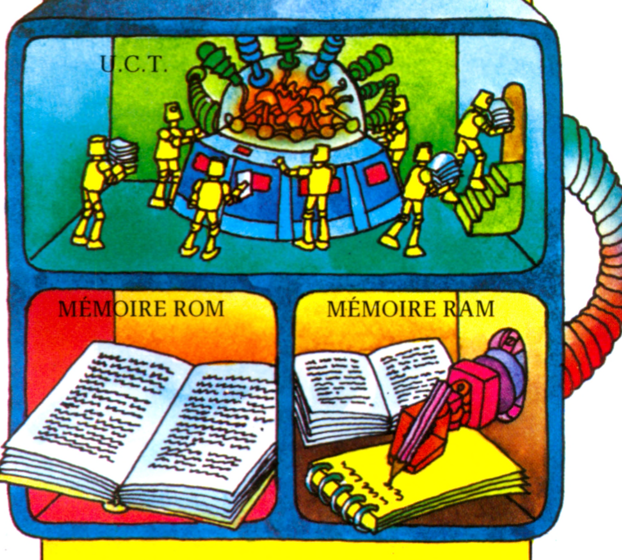
Les micro-ordinateurs

La plupart se présentent sous la forme d'une console qui ressemble au clavier d'une machine à écrire, à raccorder à un écran. Instructions et informations sont transmises à l'ordinateur en tapant sur le clavier de la console. Tout ce qui est frappé sur le clavier, ainsi que les résultats fournis par l'ordinateur, s'affichent à l'écran.

Certains micro-ordinateurs ont de petits écrans incorporés, tout comme les calculateurs de poche. D'autres peuvent être reliés à des écrans spéciaux : les *moniteurs* qui se présentent comme des téléviseurs privés de leurs circuits de réception des émissions TV. Le clavier d'un micro-ordinateur ressemble à celui d'une machine à écrire, avec quelques touches supplémentaires. Sur certains appareils, chaque touche correspond à

A l'intérieur d'un ordinateur

Un micro-ordinateur est constitué de deux « organes » essentiels : l'*unité centrale de traitement,* le cœur de l'ordinateur, où s'effectue tout le travail, et la *mémoire* où programmes et données sont stockés.



En réalité, l'ordinateur a deux mémoires. L'une, appelée ROM (Read Only Memory = mémoire en lecture seulement), contient un programme permanent. L'autre, appelée RAM (Random Access Memory = mémoire à accès direct), dont le contenu s'efface à chaque fois qu'on débranche l'ordinateur.



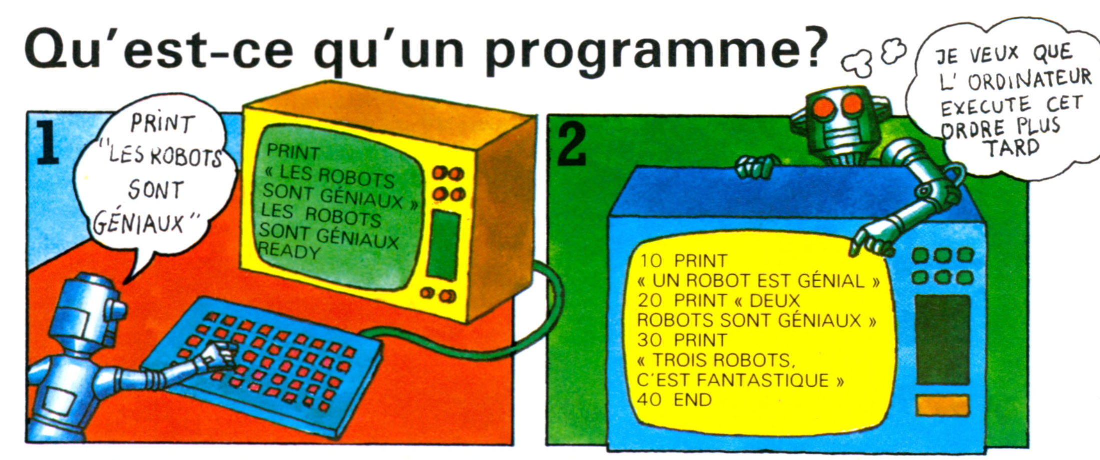


Imprimante

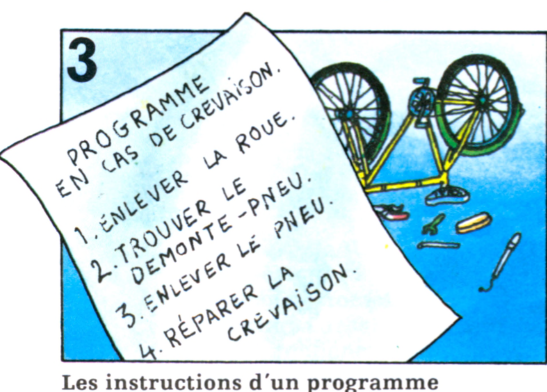
une instruction en BASIC, que vous n'aurez donc pas besoin d'entrer lettre à lettre.

L'affichage des informations s'effectue sur un écran TV ou sur un moniteur. Si l'on veut conserver une trace écrite des programmes et des données, il faut utiliser une imprimante pour les transférer sur papier. On peut également les enregistrer sur une cassette de magnétophone. Il suffira de charger celle-ci sur l'ordinateur pour retrouver les informations.

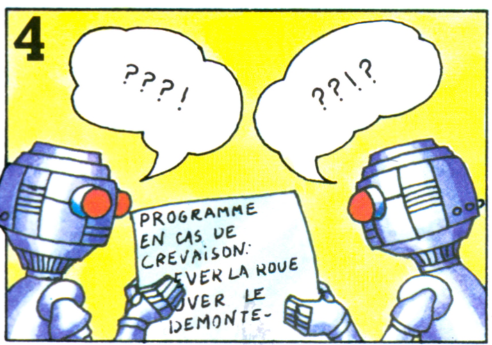
Qu'est-ce qu'un programme?



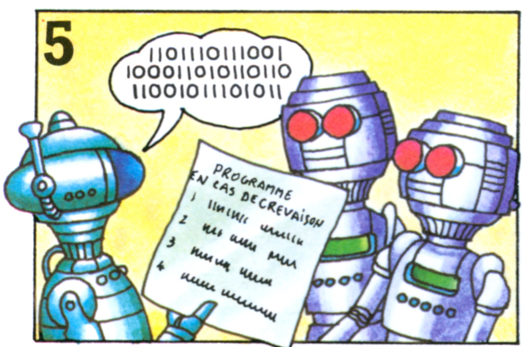
Pour faire agir l'ordinateur, il faut lui donner un ordre qu'il comprenne. Cette instruction peut être un ordre direct qu'il exécute immédiatement, ou un programme d'instructions qu'il garde en mémoire et n'exécutera que lorsque vous lui en donnerez le signal.



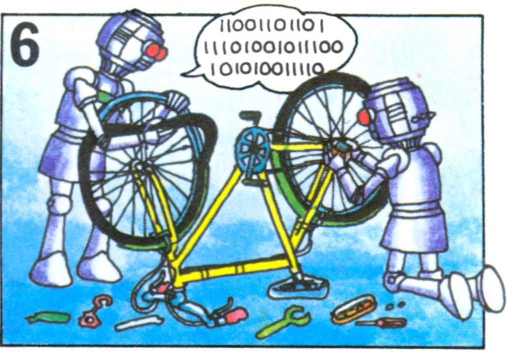
Les instructions d'un programme doivent être très soigneusement formulées. L'ordinateur essaiera d'exécuter vos instructions avec précision, même si elles sont fausses.



L'ordinateur ne comprend pas les ordres donnés dans notre langue. Il faut donc les traduire dans un des nombreux langages informatiques. Vous en trouverez quelques exemples page ci-contre.



À l'intérieur de l'ordinateur, toutes les informations circulent sous la forme d'impulsions électriques codées. Les instructions que vous donnez sont traduites dans le code de l'ordinateur par un programme spécial : l'*interpréteur* intégré à l'ordinateur.

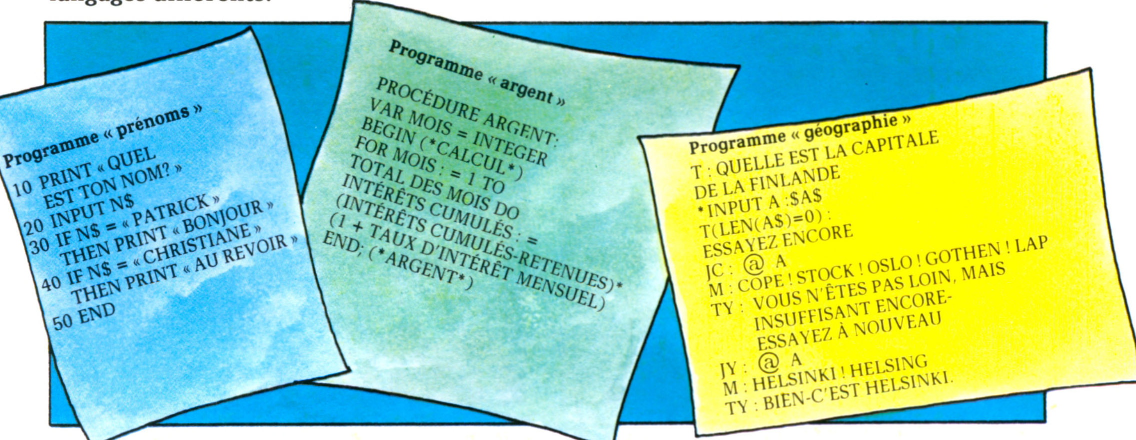


Dans ce code, chaque instruction correspond à une suite d'impulsions particulières : la valeur 1 représente une impulsion alors que la valeur 0 en marque l'absence.

Les langages de programmation

Vous pourriez écrire vos programmes directement dans le code de l'ordinateur. Mais ce serait très difficile. En revanche, il existe des langages de programmation, dits *langages évolués,* que l'ordinateur peut traduire dans sa propre langue.

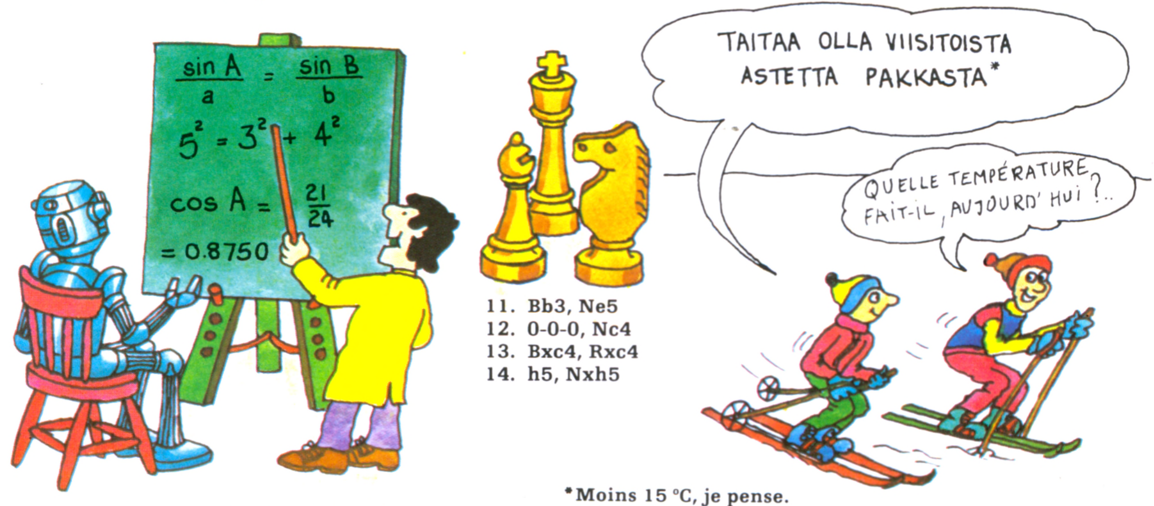
On trouve des centaines de langages évolués. La plupart ont une application dans un domaine particulier : sciences et techniques, administration ou commercial... Le BASIC est un des langages les plus courants. Le sigle BASIC signifie : Beginner's All-purpose Symbolic Instruction Code (langage général de programmation pour débutants). Mais il n'y a pas que les débutants qui l'utilisent. Vous trouverez ci-dessous des exemples de trois langages différents.



Voici un programme très court rédigé en BASIC. La ligne 10 dit à l'ordinateur d'afficher à l'écran « Quel est ton nom? ». Puis l'ordinateur entre votre réponse dans sa mémoire et, si votre nom est Patrick ou Christiane, vous transmet un message.

Ce programme est écrit en Pascal, du nom du célèbre mathématicien français. Certains pensent qu'il est plus facile d'écrire de bons programmes en Pascal qu'en BASIC.

Voici le langage PILOT. On l'utilise pour écrire des programmes pédagogiques. Avec ce langage, l'ordinateur peut reconnaître une réponse, même si elle n'est pas parfaitement exacte.



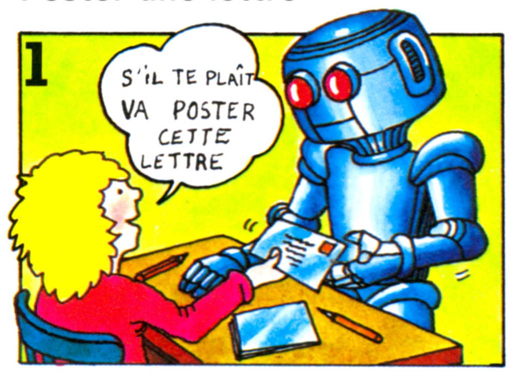
*Moins 15 °C, je pense.

A première vue, les langages de programmation semblent très compliqués, tout comme des langues étrangères tel le chinois ou le finnois que vous pouvez lire sur l'image de droite. Mais il suffit d'apprendre à les connaître. Dans beaucoup d'autres domaines aussi, on se sert de codes spéciaux : par exemple, en mathématiques on a recours à des signes particuliers pour désigner des idées et des formules et éviter une terminologie fastidieuse; le jeu d'échecs et la musique utilisent également leurs propres signes.

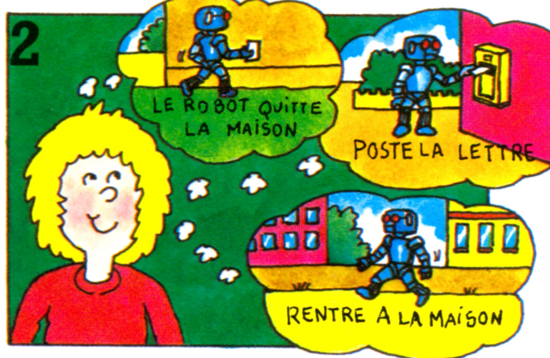
Écrire un programme

Un programme, c'est comme une règle du jeu ou une recette de cuisine. S'il y a une erreur dans la règle ou dans la recette, on ne pourra ni bien jouer, ni faire un bon gâteau. De même, les résultats qu'on obtient de l'ordinateur dépendent de la qualité des instructions qu'on lui a données. Avant d'écrire un programme, il faut d'abord étudier soigneusement ce que l'on veut faire, déterminer les étapes principales qui conduiront au résultat souhaité.

Poster une lettre



Imaginez que vous essayez d'écrire un programme qui demande à un robot de poster une lettre. Un ordre simple, comme celui exprimé dans la bulle, serait trop difficile à comprendre pour un robot.



Vous devez relevez avec précision toutes les actions que le robot doit exécuter pour poster la lettre. L'ordinateur placé à l'intérieur du robot réclame des instructions pour savoir quoi faire à chaque étape.

3

Quitter la maison
aller jusqu'à la porte
ouvrir la porte
sortir et fermer la porte
trouver la boîte aux lettres

Poster la lettre
introduire la lettre
dans la boîte
lâcher la lettre

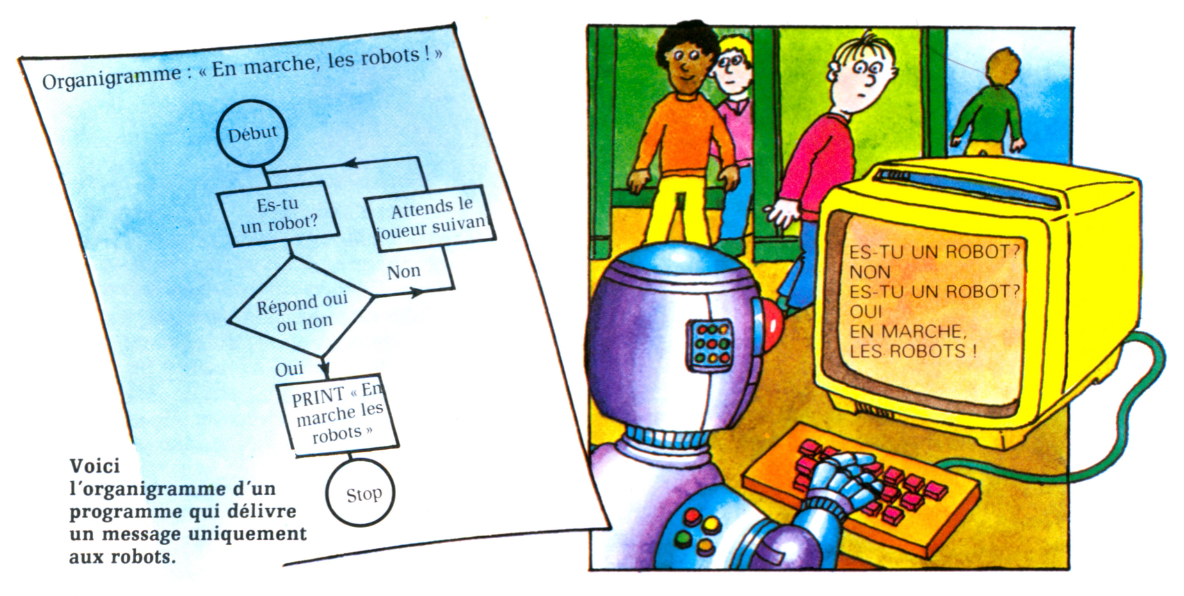
Rentrer à la maison
se retourner
revenir sur ses pas
ouvrir la porte
entrer
fermer la porte

JE DOIS POSTER CETTE LETTRE

Pour programmer, il faut découper à chaque étape les instructions en ordres simples, immédiatement traduisibles dans la langue du robot. Le robot essaiera d'exécuter vos instructions, même si elles sont mauvaises ou incomplètes. On appelle « bugs » les erreurs de programmation qui peuvent parfois conduire à des résultats inattendus de la part de l'ordinateur. Passer à travers un mur, par exemple !

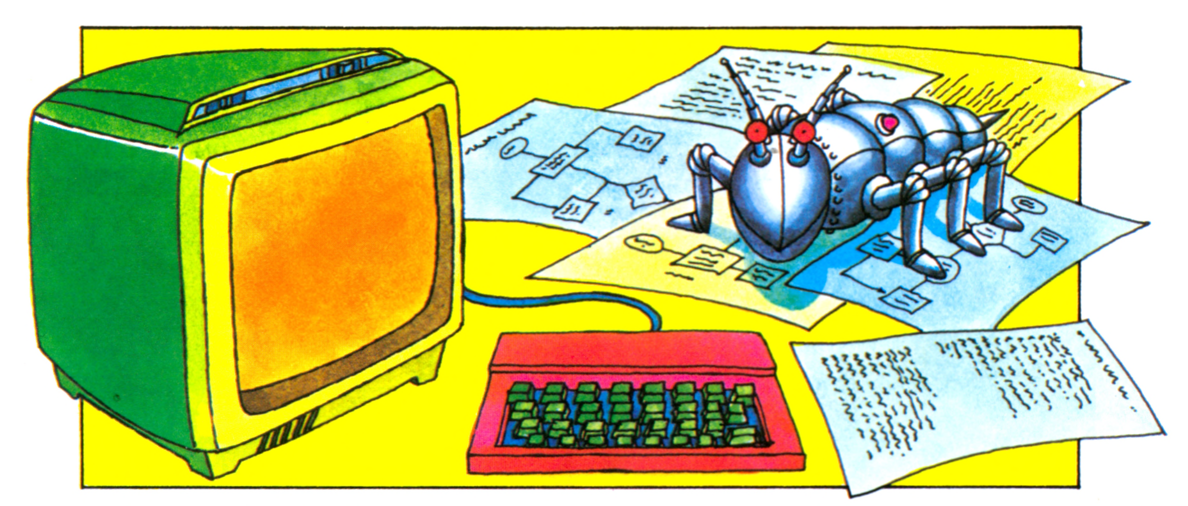
Les organigrammes

Quand on écrit un programme, il est parfois utile d'établir un schéma, ou organigramme, indiquant les principales étapes nécessaires à la résolution du problème envisagé. On y situe très exactement chaque étape du déroulement du programme.



Voici l'organigramme d'un programme qui délivre un message uniquement aux robots.

Ces diverses étapes sont indiquées par des encadrés de formes différentes. Début et fin s'inscrivent dans des formes rondes; les ordres d'exécution dans des rectangles; les prises de décision (là où l'ordinateur agit différemment selon les réponses qu'il reçoit) dans des losanges. Les traits et les flèches indiquent les itinéraires que l'ordinateur peut suivre.



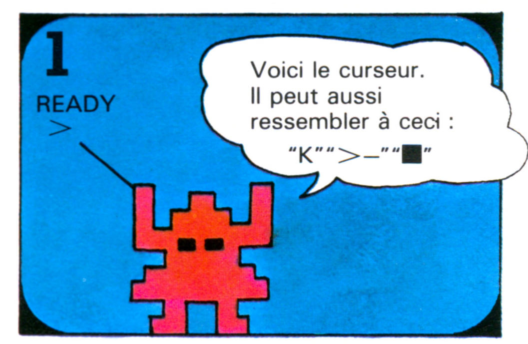
Après avoir mis en évidence tous les détails du programme, on peut le convertir en BASIC. Toutefois, il ne fonctionnera certainement pas du premier coup, car quelques bugs s'y seront sans doute glissés : erreurs de frappe faites au moment où l'on a entré le programme, erreurs de syntaxe ou erreurs de logique. Avant de faire tourner le programme, il faut détecter tous les bugs et les corriger. Parfois, un bug peut opérer une légère transformation et aboutir à un résultat que l'on aurait préféré. De tels bugs, utiles, sont appelés « pugs ».

*Vous trouverez quelques trucs pages 42 et 43 pour vous aider à trouver les bugs.

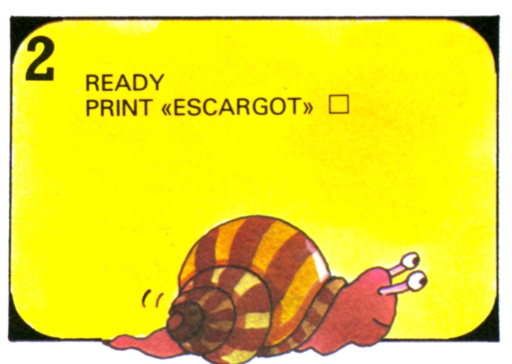
Premiers pas en BASIC

La plupart des termes utilisés en BASIC viennent de l'anglais. Ainsi « PRINT » signifie « afficher sur l'écran », « RUN » donne l'ordre de mise en route du programme, « INPUT » permet de donner une information à l'ordinateur. Ces deux pages vous expliquent comment utiliser la fonction PRINT.

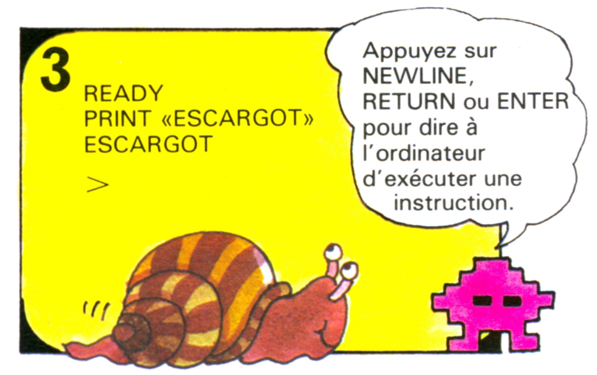
La plupart des ordinateurs domestiques possèdent un interpréteur BASIC qui leur est intégré. Aussi, une fois branchés, ils sont immédiatement prêts à être programmés dans ce langage*.



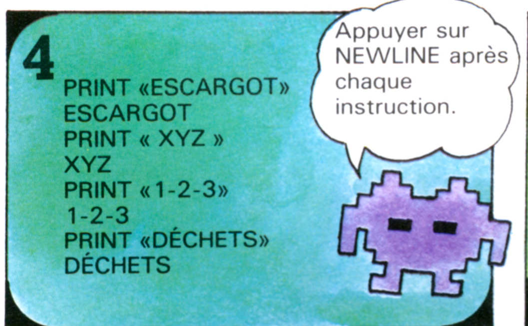
Lorsque l'on met l'ordinateur en marche, quelques mots s'affichent généralement à l'écran, accompagnés d'un petit symbole appelé curseur. Ce curseur indique à quel endroit va s'afficher la prochaine lettre que vous allez taper.



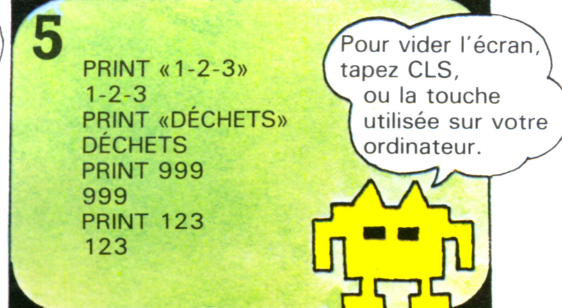
Pour demander à l'ordinateur d'afficher un mot à l'écran, on utilise la fonction PRINT, suivie du mot souhaité entre guillemets. Par exemple, PRINT « ESCARGOT » indique à l'ordinateur d'afficher le mot ESCARGOT à l'écran.



L'ordinateur n'exécutera pas votre instruction si l'on n'appuie pas sur la touche NEWLINE, RETURN ou ENTER – selon les appareils –, ce qui lui montre que l'ordre est terminé.



L'ordinateur affiche à l'écran tout ce que vous aurez placé entre guillemets, que ce soit des lettres, des chiffres, des mots ou des symboles. Remarquez qu'il n'affiche pas les guillemets eux-mêmes.

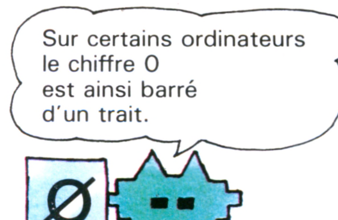


Si l'on a à afficher des nombres, il n'est pas nécessaire d'utiliser de guillemets. Pour vider l'écran, il suffit, sur la plupart des ordinateurs, de taper CLS (regardez dans votre mode d'emploi, si vous avez un ordinateur).

*Sur certains ordinateurs, il faut d'abord charger une cassette spéciale pour pouvoir programmer en BASIC.

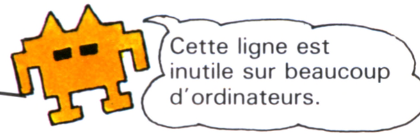
Un programme en BASIC

Dans un programme, chaque ligne d'instructions est précédée d'un numéro. L'ordinateur sait ainsi qu'il doit stocker les instructions en mémoire et ne pas démarrer avant qu'on ne lui en donne le signal. Sur la page ci-contre, aucun numéro ne figurait devant les instructions, si bien que l'ordinateur les exécutait immédiatement. Voici un court programme qui permet d'afficher à l'écran une forme de tête figurée par des symboles.




```
10 PRINT "//////"
20 PRINT "I      I"
30 PRINT "I(..)I"
40 PRINT "I -L I"
50 PRINT "VVVV"
60 END
```

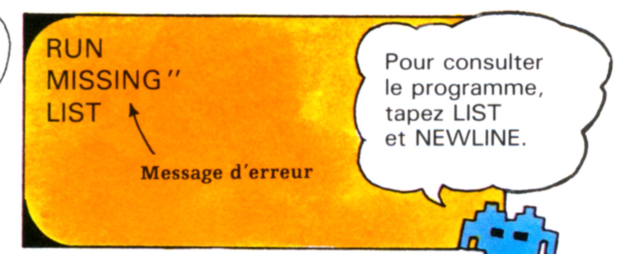
Les lignes de programme sont généralement numérotées de 10 en 10 pour qu'il soit possible d'intégrer de nouvelles instructions sans avoir tout à renuméroter.



Quand on entre un programme, on doit taper NEWLINE (ou le mot correspondant sur l'ordinateur) à la fin de chaque ligne. Les instructions sont affichées à l'écran, mais l'ordinateur ne les exécute pas tant qu'on ne lui en a pas donné l'ordre en tapant RUN. Attention à ne pas confondre la lettre O et le zéro, ce qui provoquerait un bug. Il faut être vigilant et ne pas hésiter à utiliser la touche RUBOUT (effacement) ou DELETE (suppression) qui permettent de corriger les fautes de frappe.

```
RUN
//////
I    I
I(..)I
I-LI
VVVV
```

L'ordinateur affiche exactement ce que vous avez tapé entre guillemets, y compris les espaces.

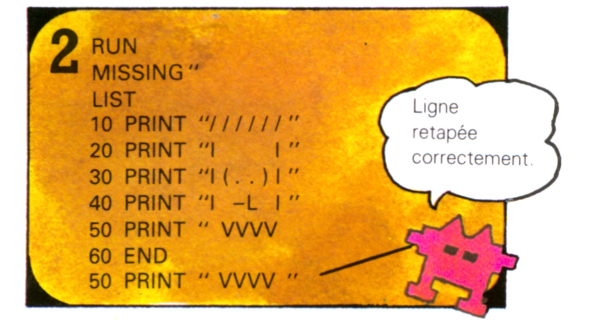


Une fois les instructions entrées, vérifiez-les soigneusement pour contrôler qu'il ne reste pas d'erreurs. Puis, pour demander à l'ordinateur d'exécuter le programme, tapez RUN suivi de NEWLINE. Si le programme ne fonctionne pas, ou si le résultat ne semble pas satisfaisant, il faut à nouveau afficher le programme pour découvrir le bug. Dans ce cas, tapez LIST. Il se peut alors que l'ordinateur indique l'erreur commise.

1 Pour dépister les bugs

```
RUN
MISSING "
LIST
10 PRINT "//////"
20 PRINT "I      I"
30 PRINT "I(..)I"
40 PRINT "I  -L  I"
50 PRINT " VVVV
60 END
```

Guillemets oubliés



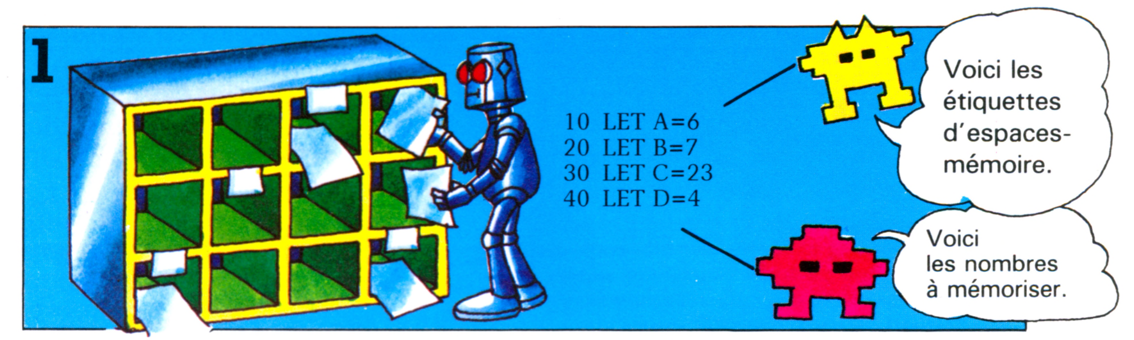
Pour la plupart des bugs, l'ordinateur précisera de quel type d'erreur il s'agit. On trouvera dans le mode d'emploi du micro-ordinateur, la signification des messages délivrés ainsi que les méthodes préconisées pour corriger ou transformer tout ou partie des lignes d'instructions. La façon la plus simple de corriger une erreur est de réécrire la ligne en entier. L'ordinateur remplacera l'ancienne ligne d'instructions par une nouvelle. Pour faire disparaître la ligne fausse, il suffit de taper le numéro de cette ligne, puis NEWLINE.



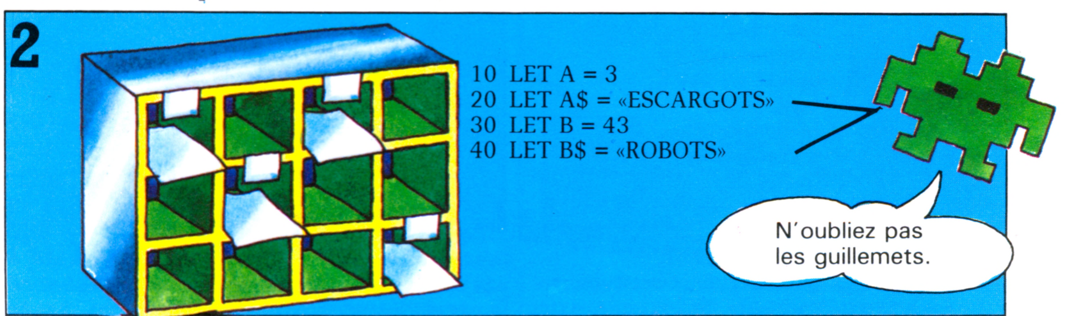
Casse-Tête – Changez de programme pour modifier des traits du personnage dessiné.

Entrer des données

Pour que l'ordinateur accomplisse des tâches plus intéressantes que d'afficher simplement des signes à l'écran, il faut lui fournir des informations qu'il puisse utiliser en les mettant en mémoire jusqu'à ce que vous lui donniez l'ordre d'aller les chercher.



Lorsque l'on introduit une donnée dans l'ordinateur, il faut lui attribuer une étiquette pour pouvoir la retrouver. A cette fin, on peut utiliser des lettres de l'alphabet. Pour repérer une case-mémoire et y inscrire un nombre, on se sert de l'instruction LET. Un espace-mémoire réservé s'appelle une *variable,* car il peut contenir des données diverses à des moments différents du programme.

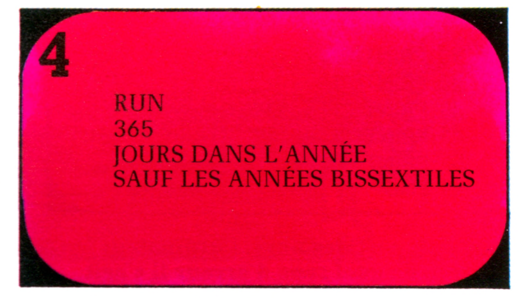


Il faut utiliser des étiquettes particulières pour mémoriser les lettres et les symboles. Ceux-ci forment une succession de caractères qu'on appelle des *chaînes alphanumériques.*
Pour les étiqueter, on a recours à une lettre de l'alphabet suivie du signe $. L'instruction LET permet de mettre une chaîne en mémoire, tout comme une variable, si ce n'est que lettres et symboles doivent être écrits entre guillemets.

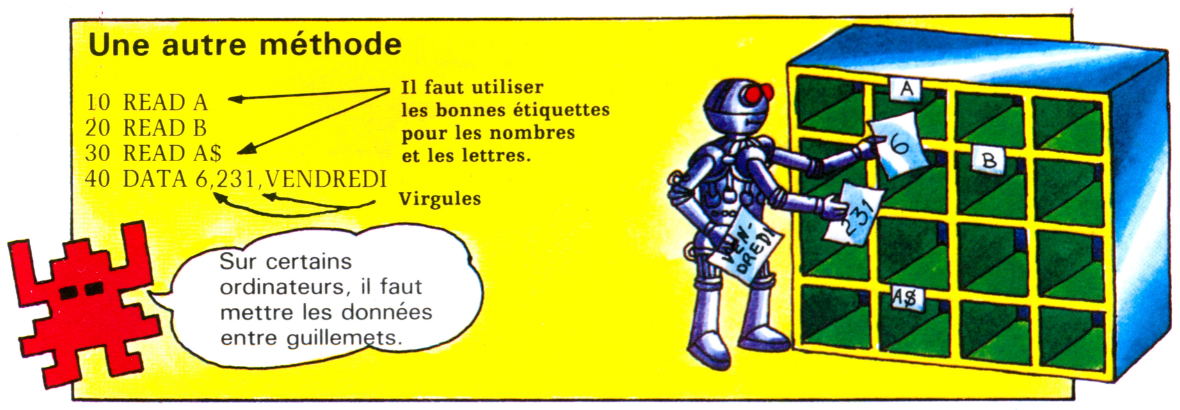
3

```
10 LET B = 365
20 LET D$ = « JOURS DANS
   UNE ANNÉE »
30 LET L$ = « SAUF LES ANNÉES
   BISSEXTILES »
40 PRINT B
50 PRINT D$
60 PRINT L$
70 END ← Beaucoup d'ordinateurs n'ont
         pas besoin de la ligne END.
```

Pour afficher l'information à l'écran, utilisez PRINT suivi du nom de la variable, par exemple PRINT A$. Ce court programme permet d'afficher le contenu des variables B, D$ et L$.



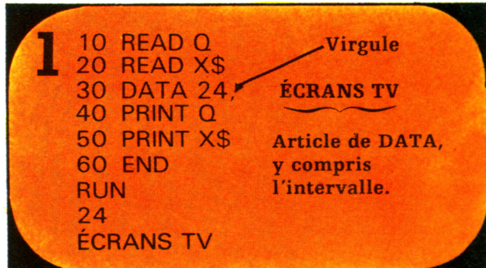
Vous pouvez relancer le programme autant de fois que vous le souhaitez. Jusqu'à ce que vous changiez les données en variable, l'ordinateur reproduira les mêmes informations.

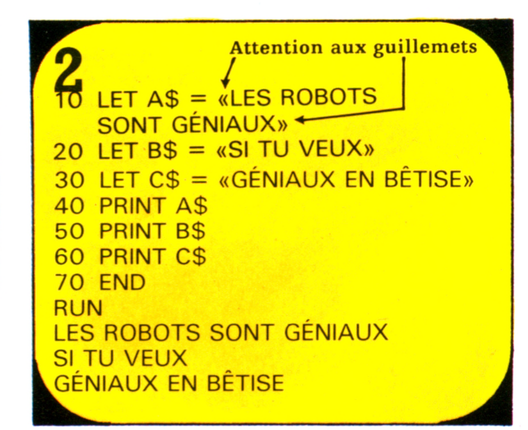


On peut aussi mémoriser l'information avec les instructions READ ou DATA. Les lignes READ indiquent à l'ordinateur d'étiqueter des espaces-mémoire, et la ligne DATA contient les informations.

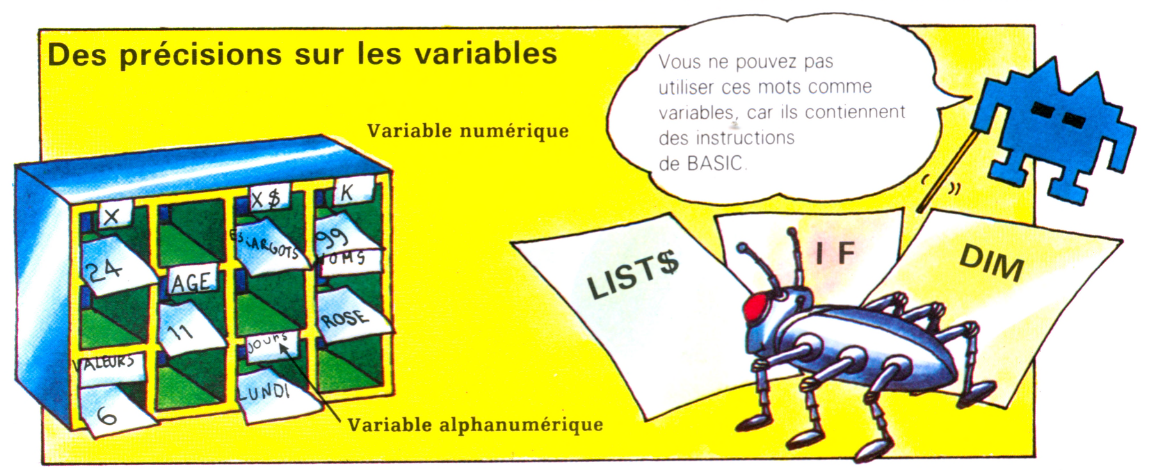
Quand on lance le programme, l'ordinateur place chaque donnée dans l'espace-mémoire correspondant. Les articles des données doivent être séparés par des virgules afin que la machine les situe*.

Quelques programmes





Voici deux programmes utilisant l'un READ et DATA, l'autre LET, pour stocker l'information.

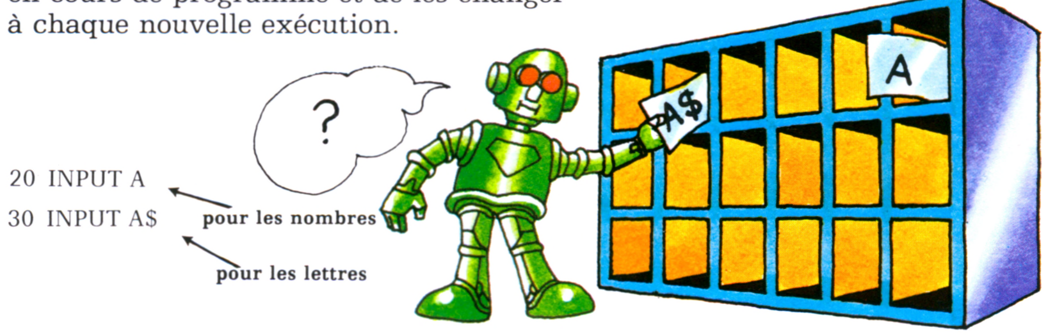


Les variables sont des espaces-mémoire étiquetés où l'information est stockée et dont le contenu peut changer en cours de programme. Une variable contenant un chiffre s'appelle une *variable numérique,* une variable comportant des lettres et des symboles, une *variable alphanumérique.* Certains ordinateurs se servent de mots comme « étiquettes de variables », à l'exception des termes de BASIC, car il y aurait alors confusion pour l'ordinateur.

*Vous ne pouvez pas employer cette méthode sur le SINCLAIR ZX81.

Comment utiliser INPUT

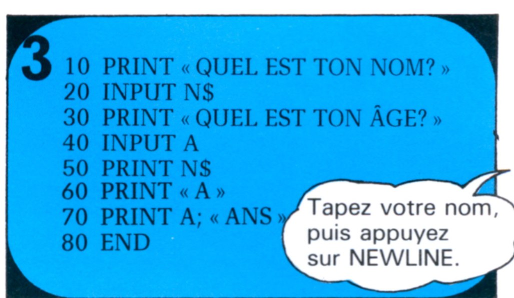
On peut encore fournir des informations à l'ordinateur par l'instruction INPUT. Ce qui permet d'introduire des données en cours de programme et de les changer à chaque nouvelle exécution.

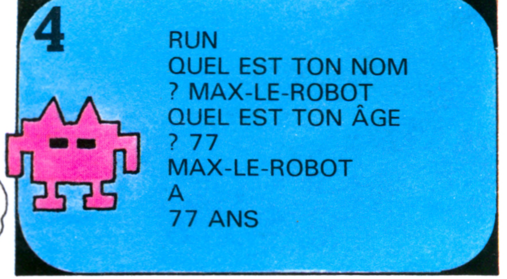


On utilise INPUT avec une étiquette comme A pour un nombre et A$ pour une lettre. Lorsque l'ordinateur rencontre l'instruction INPUT dans le déroulement d'un programme, il place l'étiquette dans un espace-mémoire et demande quelle est la donnée, en inscrivant un point d'interrogation, ou tout autre symbole, sur l'écran. On entre alors la donnée, que l'ordinateur mémorise, avant de continuer à exécuter le programme.



La Figure 2 montre ce qui se passe lorsque vous lancez ce programme. Quand l'ordinateur rencontre INPUT à la ligne 10, il inscrit un point d'interrogation et attend qu'on lui indique la valeur de G. Puis il affiche à nouveau un point d'interrogation pour l'INPUT de la ligne 40. Cette fois, il faut entrer des mots ou des symboles, car l'étiquette B$ indique à l'ordinateur qu'il doit attendre une chaîne de caractères.





Si vous avez un ordinateur, essayez d'entrer ce programme, puis tapez RUN pour le lancer. Lorsque l'ordinateur vous demandera des informations, donnez-lui votre nom et votre âge, ou n'importe quel nom et chiffre fantaisistes comme dans l'exemple donné ci-dessus. Renouvelez l'expérience plusieurs fois en introduisant des données différentes. L'ordinateur affichera toujours ce que vous venez d'entrer en N$ et A.

Un peu de poésie !

Vous savez maintenant assez de BASIC pour écrire un poème sur un ordinateur. Voici un programme de poésie qui utilise les instructions PRINT et INPUT.

```
10 PRINT «QUEL EST TON NOM»
20 INPUT N$
30 PRINT «POÈME ÉCRIT PAR»
40 PRINT N$
50 PRINT «TAPE UN MOT»
60 PRINT «QUI RIME AVEC PEUR»
70 INPUT A$
80 PRINT «VOICI LE POÈME»
90 PRINT «DANS LE TEMPS LES ORDINATEURS
   ME FAISAIENT PEUR»
100 PRINT «MAIS MAINTENANT JE SUIS
    AUSSI HEUREUX QU'UN»
110 PRINT A$
120 END
```

A cette ligne s'inscrit votre nom

A cette ligne s'inscrit le mot que vous avez choisi.

```
RUN
QUEL EST TON NOM
? JEAN
POÈME ÉCRIT PAR JEAN
TAPE UN MOT
QUI RIME AVEC PEUR
? RATON LAVEUR
VOICI LE POÈME
DANS LE TEMPS LES
ORDINATEURS ME FAISAIENT PEUR
MAIS MAINTENANT JE SUIS AUSSI
HEUREUX QU'UN RATON LAVEUR
```

Votre nom et le mot.

Tapez RUN pour recommencer avec un autre mot.

En exécutant le programme, l'ordinateur vous demande votre nom, mémorise votre réponse en N$, puis l'affiche à la ligne 40. Il stocke le mot que vous avez choisi en A$, et l'écrit à la ligne 110, comme s'il était intégré au poème. Si vous avez un ordinateur, testez ce programme plusieurs fois en entrant des mots différents à la ligne 70.

Casse-tête

Écrivez un programme dans lequel l'ordinateur vous demande votre nom, puis écrit Bonjour, suivi de votre nom et d'un message à votre intention.

Solution page 44.

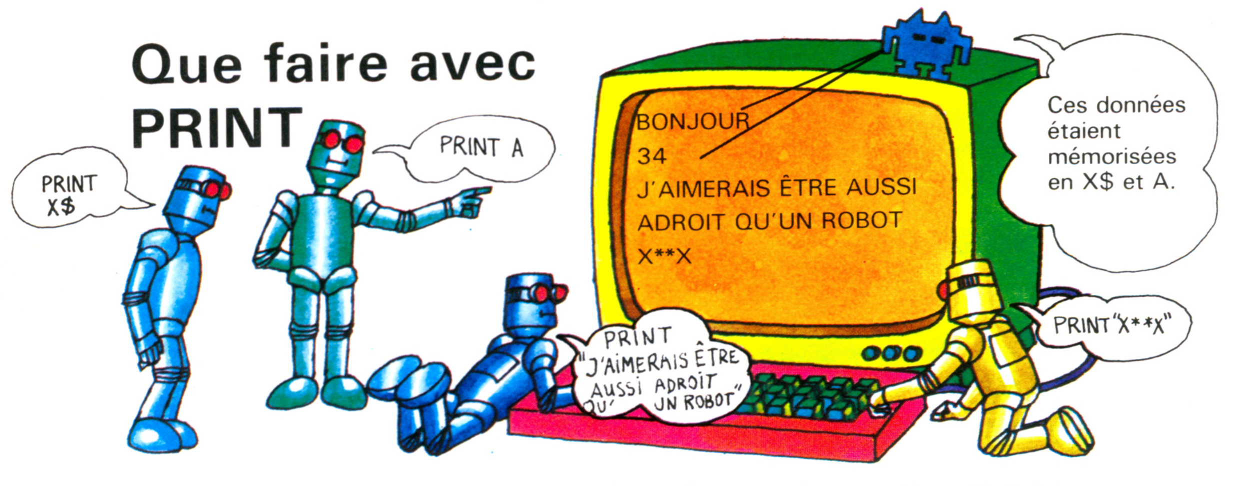
Ce qu'il faut faire avant d'entrer un programme

1. **Avant d'entrer un nouveau programme, tapez NEW. Vous effacerez ainsi tous les listings ou variables restant en mémoire.**
2. **Pendant que vous entrez le programme, n'oubliez pas de taper sur la touche NEWLINE (ou la touche correspondante de votre ordinateur) à la fin de chaque ligne d'instructions.**
3. **Après avoir entré le programme, vérifiez chaque ligne pour contrôler qu'il n'y a pas de fautes de frappe ni de ligne oubliée.**
4. **Vous pouvez alors taper CLS (ou l'instruction propre à votre ordinateur) pour vider l'écran. Lancez ensuite le programme en tapant RUN.**
5. **Pour faire réapparaître le programme, que ce soit pour le vérifier ou le modifier, tapez LIST. Pour visualiser une ligne en particulier, vous pouvez taper LIST suivi du numéro de la ligne d'instructions. Contrôlez sur la notice d'utilisation de votre micro-ordinateur.**
6. **Pour arrêter le programme en cours d'exécution, tapez BREAK ou ESCAPE. Là aussi, vérifiez dans votre notice. Attention sur certains appareils, ESCAPE élimine le programme de la mémoire de l'ordinateur. Pour relancer le programme, tapez RUN.**

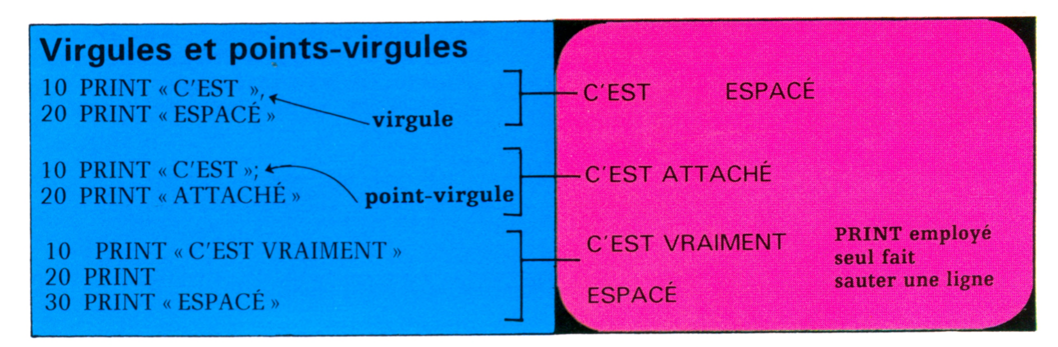
Voir pages 42-43 comment éviter quelques bugs.



Que faire avec PRINT

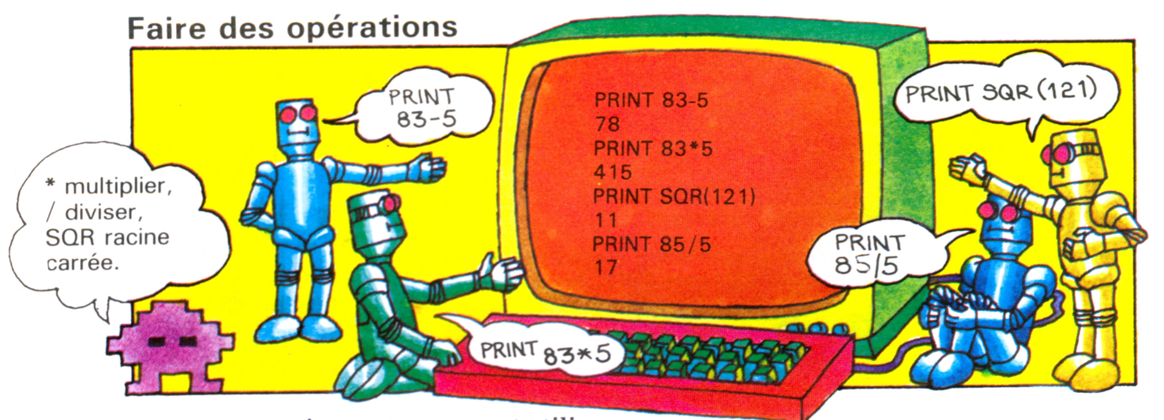


Jusqu'à présent, on a vu comment utiliser PRINT pour afficher des chiffres et des lettres à l'écran, ou pour visualiser des variables. Nous allons maintenant apprendre à jouer avec les virgules et les points-virgules pour situer l'affichage sur l'écran. Avec PRINT, on peut aussi effectuer des opérations. La page ci-contre donne d'autres exemples d'utilisation des variables.



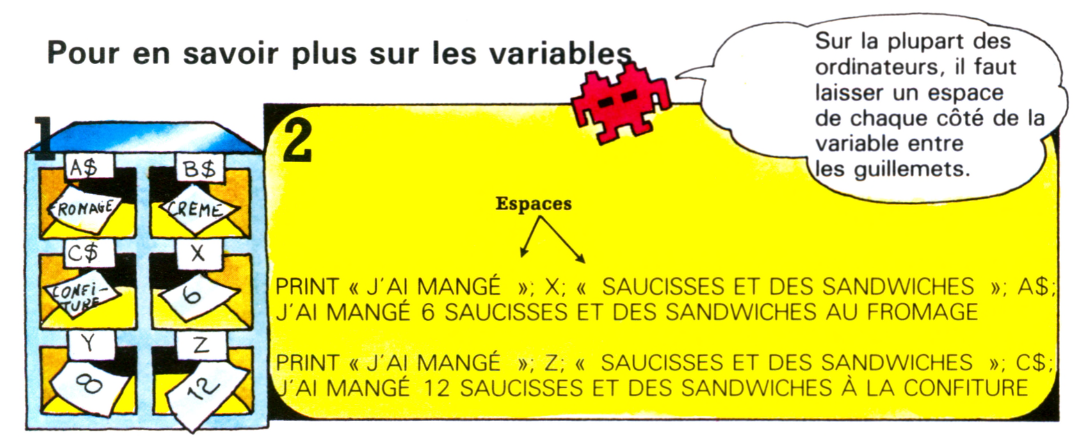
Voici comment utiliser les virgules et les points-virgules pour indiquer à l'ordinateur où il doit afficher la donnée. Une virgule lui indique de laisser un espace; un point-virgule signifie qu'il faut enchaîner directement. L'image montre comment s'afficheraient les lignes sur l'écran selon le signe de ponctuation choisi. L'instruction PRINT employée seule fait sauter une ligne.

Faire des opérations

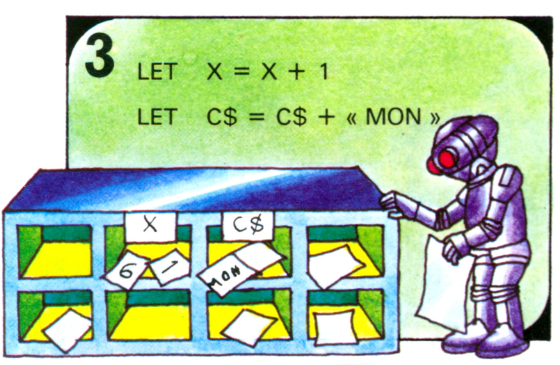


Vous savez, maintenant, comment utiliser PRINT. Pour l'addition et la soustraction, servez-vous des signes + et − habituels; pour multiplier, tapez* (pour ne pas confondre ce signe avec la lettre x), et pour diviser/. L'ordinateur peut réaliser des opérations plus sophistiquées comme le calcul des sinus, cosinus, racines carrées, etc.

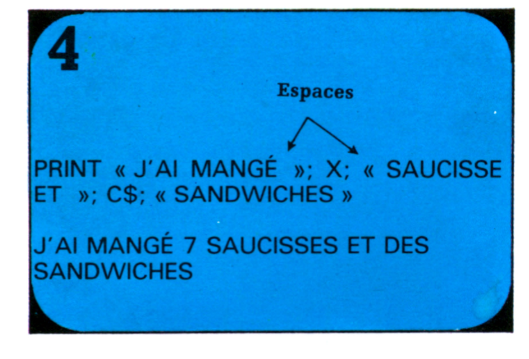
Pour en savoir plus sur les variables



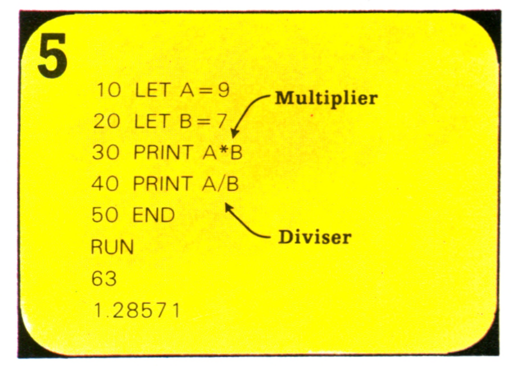
Afficher des variables n'est pas, en soi, très intéressant. Il faut souvent les intégrer dans une phrase. Pour afficher des mots et une variable ensemble, on doit placer les mots entre guillemets et mettre un point-virgule de part et d'autre de la variable. Si l'on veut que l'information se détache, il suffit d'utiliser des virgules à la place de points-virgules.



En cours de programme, on peut changer le contenu des espaces-mémoire. Pour l'ordinateur, l'exemple cité dans ce dessin signifie qu'il faut ajouter 1 au chiffre étiqueté X et « mon » aux lettres étiquetées C$.



Chaque fois que l'on demande à l'ordinateur d'afficher les variables, il restitue les dernières données enregistrées.



On peut également effectuer des opérations avec des variables. L'ordinateur va chercher les nombres dans les espaces-mémoire, puis donne le résultat.

Casse-tête

1. Écrivez un programme qui permette d'ajouter des nombres aux variables du programme situé à gauche de telle sorte que les réponses 100 et 1 s'affichent sur une même ligne, tout en étant séparées par un espace.

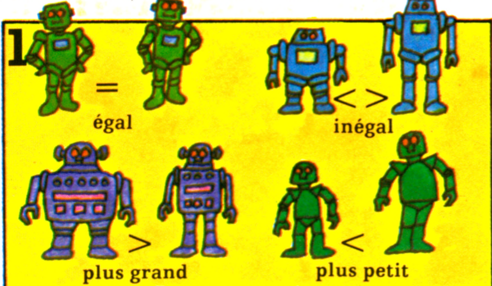
2. Transformez les lignes 30 et 40 pour qu'elles inscrivent les nombres, les opérations effectuées et les résultats : par exemple « 7 fois 9 font 63 ».

3. Répondez au casse-tête de la page 15 de telle sorte que votre nom et le message s'inscrivent sur une même ligne.

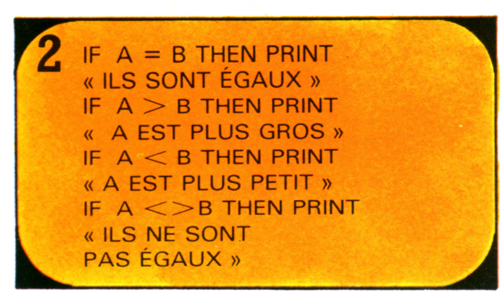
Comment les ordinateurs comparent

Une des fonctions de l'ordinateur : être capable de comparer des informations et d'en tirer les conséquences. Pour cela, on utilise les instructions IF ... THEN (si ... alors).

SI (IF) IL FAIT PLUS FROID DEMAIN ALORS (THEN) JE METTRAI MON ÉCHARPE



L'ordinateur peut réaliser plusieurs types de mesures pour comparer des informations. Vous voyez les symboles utilisés ci-dessus. Le micro-ordinateur peut déterminer si deux éléments sont identiques ou différents, si l'un est plus grand ou plus petit que l'autre.



2

```
IF A = B THEN PRINT
« ILS SONT ÉGAUX »
IF A > B THEN PRINT
« A EST PLUS GROS »
IF A < B THEN PRINT
« A EST PLUS PETIT »
IF A <>B THEN PRINT
« ILS NE SONT
PAS ÉGAUX »
```

Ces lignes montrent comment combiner les symboles avec les instructions IF et THEN pour que l'ordinateur effectue des comparaisons entre éléments. On peut comparer n'importe quels éléments : mots, chiffres ou variables.



3 Programme météo

```
10 PRINT « QUEL TEMPS FAIT-IL AUJOURD'HUI »
20 INPUT T$
30 IF T$ = « PLUVIEUX » THEN
   PRINT « PENSEZ AU PARAPLUIE »
40 IF T$ = « ENSOLEILLÉ » THEN
   PRINT « C'EST BON »
50 END
```


4

```
RUN
QUEL TEMPS FAIT-IL
AUJOURD'HUI
? ENSOLEILLÉ
C'EST BON
RUN
QUEL TEMPS FAIT-IL
AUJOURD'HUI
? PLUVIEUX
PENSEZ AU
PARAPLUIE
```

Voici un programme qui utilise IF et THEN. A la ligne 20, l'ordinateur met en mémoire le mot entré en variable T$. Puis, aux lignes 30 et 40, il cherche à savoir si le mot entré en T$ est « pluvieux » ou « ensoleillé ». Si c'est le cas, il affiche une des réponses indiquées. Sinon, rien ne se passe; il faudra alors modifier les mots des lignes 30 et 40.

5 Question d'âge

```
10 PRINT « QUEL ÂGE AS-TU? »
20 INPUT A
30 IF A > 16 THEN PRINT « VIEUX »
40 IF A < 16 THEN PRINT « JEUNE »
50 IF A = 16 THEN PRINT « C'EST BON »
RUN
QUEL ÂGE AS-TU
? 16
C'EST BON
```

6 Leçon d'anglais

```
10 PRINT « COMMENT DIT-ON ROUGE
EN ANGLAIS »
20 INPUT A$
30 IF A$ = « RED » THEN PRINT « JUSTE »
40 IF A$ <> « RED » THEN PRINT
« NON, RED »
RUN
COMMENT DIT-ON ROUGE EN ANGLAIS
? BLUE
NON, RED
```

Dans ce programme « Quel âge as-tu? », l'ordinateur compare l'INPUT A avec le chiffre 16. S'il est plus petit que 16, il affiche « jeune »; plus grand, il affichera « vieux »; s'il est égal à 16, il affiche « c'est bon ». Dans le programme de droite, l'ordinateur affiche l'une ou l'autre réponse, selon que A est identique à « red » ou non.

Casse-tête – Écrivez un programme dans lequel l'ordinateur vous propose des opérations à résoudre, puis réponde « juste » ou donne la bonne réponse.

Branchements de programmes



2

```
10   PRINT K$
20   IF K$ = « OUI » THEN
GOTO 100
30   IF K$ = « NON » THEN
GOTO 200
100    PRINT « VOUS AVEZ TAPÉ OUI »
110    STOP
200    PRINT « VOUS AVEZ TAPÉ NON »
210    END
```

Ces deux lignes envoient l'ordinateur à d'autres parties du programme.

On peut donner presque n'importe quel ordre à l'ordinateur après l'instruction THEN. Ce qui s'avère utile, c'est par exemple de l'envoyer à une autre ligne du programme (sur la plupart des ordinateurs, hormis le SINCLAIR ZX81, on peut se dispenser de l'instruction GOTO). Il faut introduire un STOP après GOTO, sinon l'ordinateur continuerait à répéter sans fin le même programme.

Programme de mathématiques

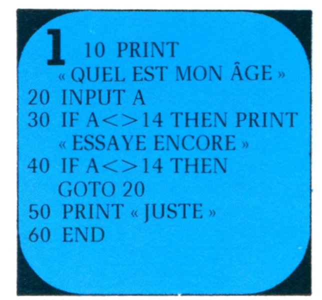
```
10   PRINT « TAPEZ UN NOMBRE »
20   INPUT A
30   PRINT « TAPEZ UN AUTRE NOMBRE »
40   INPUT B
50   PRINT « QUE VOULEZ-VOUS FAIRE »
60   PRINT « ADDITIONNER, SOUSTRAIRE,
             MULTIPLIER »
70   PRINT « DIVISER OU ARRÊTER »
80   INPUT C$
90   IF C$ = « ADDITIONNER » THEN
                    PRINT A + B
100   IF C$ = « SOUSTRAIRE » THEN
                    PRINT A – B
110   IF C$ = « MULTIPLIER » THEN
                    PRINT A*B
120   IF C$ = « DIVISER » THEN PRINT A/B
130   IF C$ = « STOP » THEN STOP
140   GOTO 10
```

```
RUN
TAPEZ UN NOMBRE
? 17
TAPEZ UN AUTRE NOMBRE
? 184
QUE VOULEZ-VOUS FAIRE
ADDITIONNER, SOUSTRAIRE, MULTIPLIER
DIVISER OU STOPPER
? ADDITIONNER
201  <----  Réponse de l'ordinateur
TAPEZ UN NOMBRE
?
```

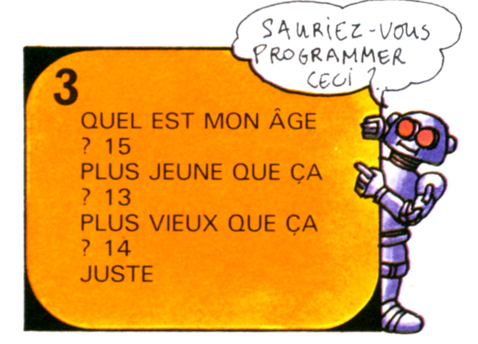
LE RROGRAMME NE S'ARRETERA QU'EN ENTRANT L'INSTRUCTION STOP

Dans ce programme, les nombres entrés sont stockés en A et B et vos ordres en C$. Des lignes 80 à 130, l'ordinateur compare C avec cinq mots différents. Quand il reconnaît le bon mot, il exécute l'instruction. Il saute toutes les lignes qui ne le concernent pas.

Programme « Quel est ton âge »







Ce programme se répètera jusqu'à ce que A = 14. Quand A = 14, l'ordinateur sautera les lignes 30 et 40 et affichera « bonne réponse ». Pouvez-vous modifier le programme de telle sorte qu'il vous fournisse quelques indications, comme le montre le dessin de droite?

Programmes en BASIC

Les programmes présentés sur ces deux pages utilisent la plupart des notions de BASIC que nous avons rencontrées. Le premier programme propose un jeu spatial sur ordinateur pour deux joueurs. Si vous n'avez pas d'ordinateur, regardez bien ces programmes et essayez de comprendre leur fonctionnement.

Commando de l'espace

```
10 PRINT « POSITION DE L'ENNEMI : LATITUDE »
20 INPUT A
30 PRINT « POSITION DE L'ENNEMI : LONGITUDE »
40 INPUT B
50 CLS
60 PRINT « POSITION DU COMMANDO : LATITUDE »
70 INPUT C
80 PRINT « POSITION DU COMMANDO : LONGITUDE »
90 INPUT D
100 CLS
110 LET X=SQR((A-C)*(A-C)+(B-D)*(B-D))
120 PRINT « VOUS ÊTES MAINTENANT A »
130 PRINT X; « CASES L'UN DE L'AUTRE »
140 IF X<1.5 THEN PRINT « ENNEMI LOCALISÉ »
150 IF X<1.5 THEN STOP
155 PRINT « INDIQUEZ VOS NOUVELLES POSITIONS »
160 GOTO 10
170 END
```

De la ligne 10 à 40, l'ordinateur mémorise les coordonnées ennemies en A et B.

La ligne 50 fait disparaître les coordonnées ennemies.

De la ligne 60 à 90, l'ordinateur mémorise les coordonnées du commando en C et D.

Cette ligne calcule la distance qui sépare les combattants et mémorise la réponse en X (ligne 110).

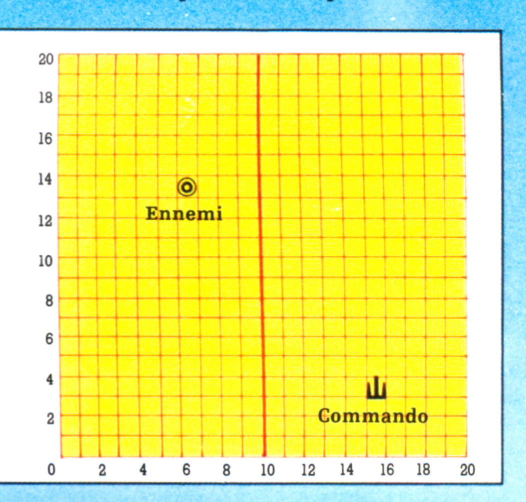
Si X est plus petit que 1,5 le jeu s'arrête. Si X est plus grand que 1,5 le programme recommence.

Dans ce jeu, un commando essaye d'attraper l'ennemi. Chaque joueur dessine une carte secrète sur laquelle il inscrit ses positions (on vous indique ci-dessous comment procéder).

Chacun transmet ses coordonnées à l'ordinateur qui détermine quelle distance sépare les adversaires. Les calculs de l'ordinateur aident les joueurs à choisir leur prochain coup.

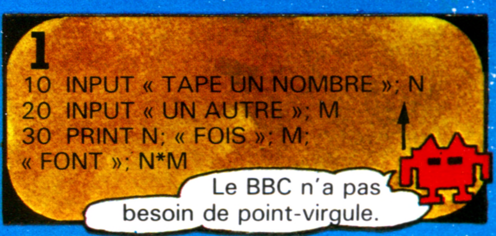
Règle du jeu

La carte secrète : chaque joueur dessine une grille de 20 carrés sur 20 carrés et les numérote comme indiqué ci-contre. Les ennemis démarrent sur la gauche de la grille, le commando sur la droite. A chaque coup, on peut se déplacer de deux cases, dans n'importe quelle direction. Puis on indique ses nouvelles coordonnées à l'ordinateur. Quand le commando parvient à moins d'1,5 cases de l'ennemi, on estime qu'il l'a attrapé.

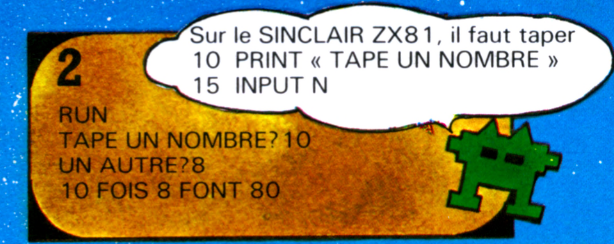


Comment donner l'air intelligent à l'ordinateur

Dans ce programme, l'ordinateur donne l'impression de dialoguer et d'établir une véritable conversation avec vous. Le fonctionnement de ce programme figure dans les dessins en bas de page. On utilise l'instruction INPUT d'une manière un peu différente de celle que nous avons décrite jusqu'ici. Ce qui rend le programme plus court et plus facile à lire.



Sur la plupart des ordinateurs (sauf le SINCLAIR ZX81), on peut rendre la ligne INPUT plus explicite en plaçant quelques mots entre guillemets avant la variable.



Dans le déroulement du programme, le point d'interrogation apparaît à la fin de la chaîne de caractères.

Programme

```
5 LET C=0
10 PRINT « J'AIMERAIS QU'ON PARLE ENSEMBLE »
20 INPUT « RACONTE-MOI QUELQUE CHOSE DE DRÔLE QUI
   TE SOIT ARRIVÉ CETTE SEMAINE »; A$
30 READ B$
40 PRINT B$   ← L'ordinateur reste à la même ligne.
50 INPUT C$   ← Votre réponse est mémorisée en C$.
60 LET C=C+1
70 IF C=6 THEN GOTO 100
80 GOTO 30
90 DATA POURQUOI, POURQUOI DONC
95 DATA EXPLIQUE-MOI, POURQUOI
98 DATA DIS-MOI POURQUOI, POUR QUELLE RAISON
100 PRINT « AINSI, LA VÉRITABLE RAISON POUR LAQUELLE »
110 PRINT «      »;A$
120 PRINT « C'EST QUE »
130 PRINT «      »;C$
140 PRINT « COMME C'EST BIZARRE »
150 PRINT « TAPE RUN, QUE NOUS CONTINUIONS
    À NOUS AMUSER »
160 END
```

Voici la nouvelle façon d'utiliser INPUT. Votre réponse est mémorisée en A$.

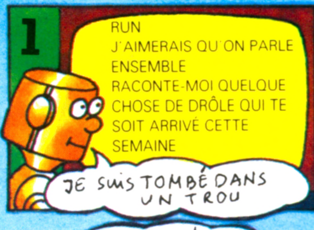
A la ligne 30, l'ordinateur recherche la première apparition de DATA, prend le premier article et l'inscrit en B$.

La variable C aux lignes 60 et 70 compte le nombre de tours effectués : elle agit comme un contrôleur de boucle. Lorsque C = 6, tous les articles en DATA ont été utilisés et l'ordinateur passe à la ligne 100.

La ligne 80 fait retourner l'ordinateur à la ligne 30; c'est l'article suivant dans la liste de DATA qui apparaîtra.

Les blancs laissés aux lignes 110 et 130 permettent de libérer un espace à l'écran avant que ne s'affiche votre réponse. Peu importe le nombre d'espacements laissés.

Comment ça marche







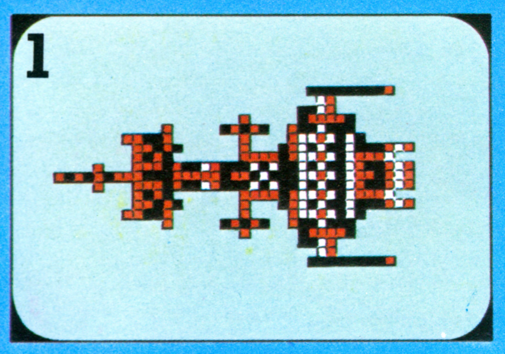




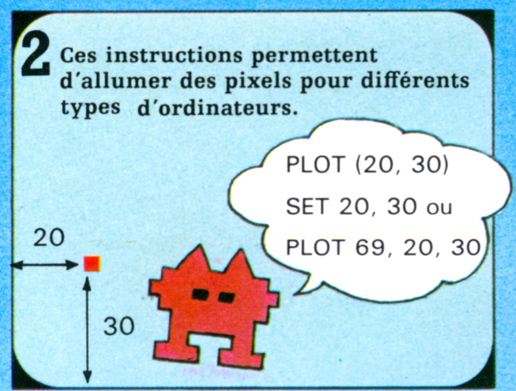
AINSI, LA VÉRITABLE RAISON POUR LAQUELLE
JE SUIS TOMBÉ DANS UN TROU
C'EST QUE JE NE VOULAIS PAS QUE MON COPAIN VOIE QUE J'AVAIS UNE GLACE
COMME C'EST BIZARRE
TAPE RUN, QUE NOUS CONTINUIONS À NOUS AMUSER

Dessinons

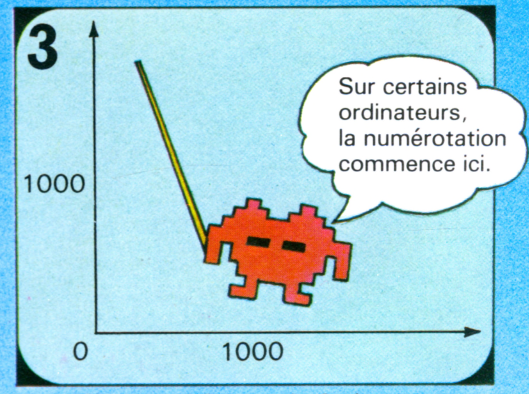
Un ordinateur compose des dessins en allumant des petits rectangles sur l'écran. Chaque rectangle, pixel en anglais, qu'on peut appeler un témoin, répond à une instruction particulière. Dans ces deux pages, on découvre comment se servir du BASIC pour dessiner sur l'écran. Beaucoup d'ordinateurs permettent de dessiner en couleurs : reportez-vous donc à votre notice explicative pour profiter de cet avantage.



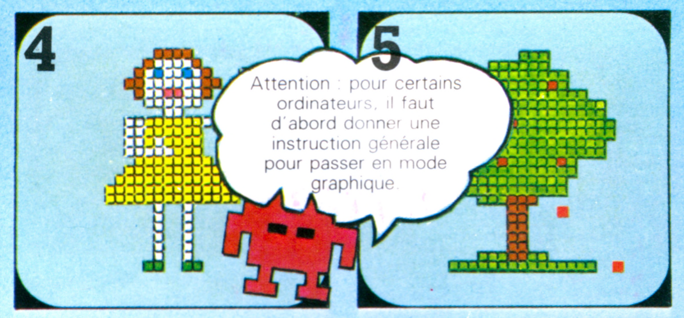
Souvent, on distingue les témoins graphiques sur le dessin. Cependant un ordinateur à grande mémoire peut réaliser des dessins avec des milliers de tout petits points. Ils sont dits dessins de très haute résolution.



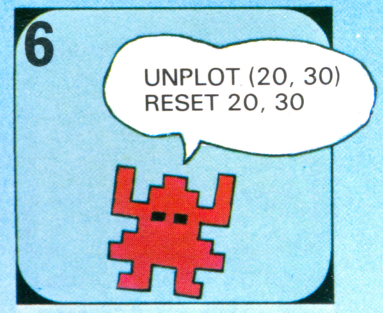
L'instruction qui permet d'allumer un pixel varie selon les appareils, mais elle se présente généralement sous la forme de PLOT (X, Y). X et Y sont les coordonnées en abscisse et en ordonnée des points témoins.



Un ordinateur à haute résolution graphique offre jusqu'à 1 000 points en abscisses et en ordonnées. Un ordinateur moins puissant présente une grille d'environ 60 × 40. (Vérifiez les dimensions de votre écran, car vous risquez de situer votre point en dehors des limites définies.)



On appelle graphiques les dessins réalisés par un ordinateur. Sur certains appareils, il faut donner une instruction spéciale pour passer en mode graphique. Par exemple, sur le BBC, on doit taper MODE plus un nombre*.



Pour éteindre un pixel, on utilise l'instruction UNPLOT (X, Y). Vérifiez cette commande dans la notice.

*** Pour les programmes de ce livre, utilisez MODE 5, puis PLOT 69, X, Y. Pour éteindre, PLOT 71, X, Y.**

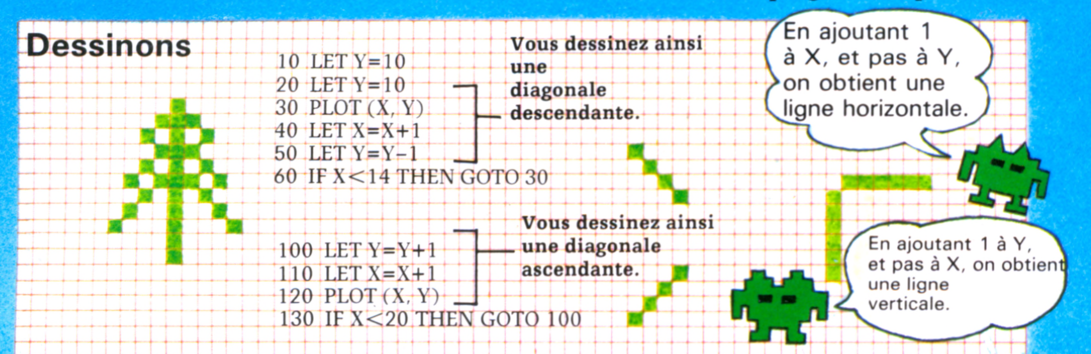
Programmes de tracés



Ce court programme réclame deux nombres, puis allume deux pixels aux coordonnées indiquées. Vérifiez que les nombres que vous entrez sont compris dans les limites de votre ordinateur.

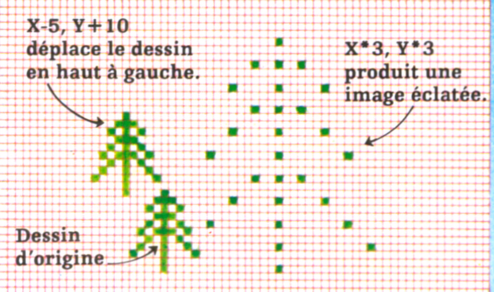
La ligne 50 permet au programme de se répéter sans fin. La seule façon de l'arrêter, c'est de taper sur la touche BREAK (ou la touche correspondante de votre ordinateur). Sauriez-vous introduire un contrôleur de boucle (cf. page 69) qui ne fasse tourner le programme que six fois ?

Dessinons



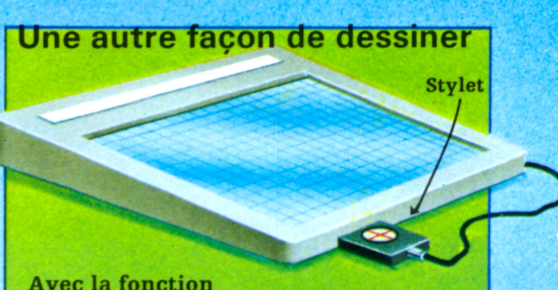
Il faut d'abord tracer le dessin sur une feuille quadrillée et déterminer les coordonnées de la figure.

On peut alors utiliser le programme pour reporter les coordonnées de chaque case. A partir des valeurs d'origine de X et Y, en les augmentant ou en les diminuant, et en faisant se répéter certaines parties du programme, l'ordinateur trace des séries de points.



Une fois le programme écrit, il est facile de transformer le dessin en changeant la valeur des nombres. On peut le déplacer sur l'écran, en modifiant les valeurs d'origine, ou produire une « image éclatée » en multipliant tous les nombres par trois.

Une autre façon de dessiner



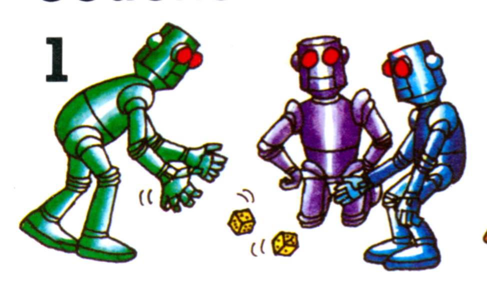
Avec la fonction PLOT, on n'obtient que des dessins rudimentaires. Pour faire mieux, il faut utiliser un matériel spécial, comme une tablette graphique. On place un dessin sur la tablette, puis on en suit le tracé avec un stylet. Les coordonnées s'enregistrent automatiquement dans l'ordinateur.

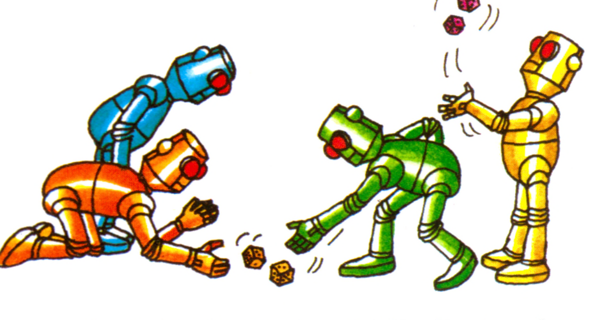


Casse-tête. Écrivez un programme qui fasse se dessiner vos initiales à l'écran. Vous en trouverez un exemple page 44.

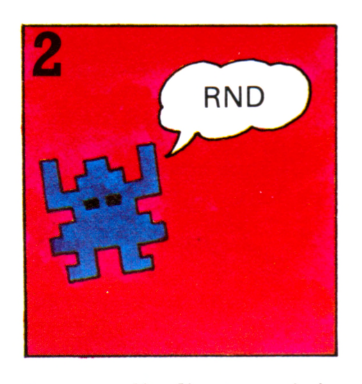
Jouons

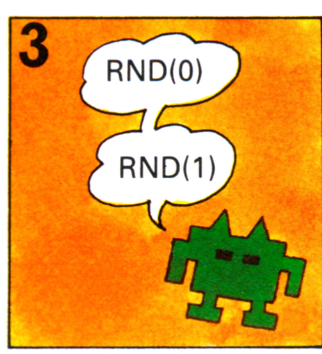
1

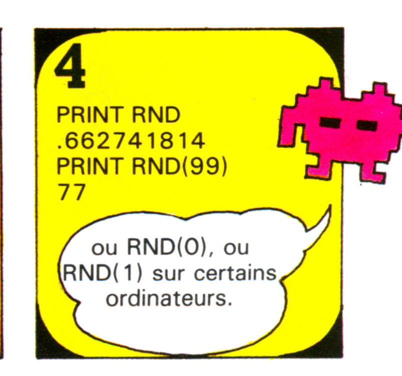




Quand on lance les dés, on ne sait jamais quels numéros sortiront. N'importe quel chiffre entre 1 et 6 a des chances égales. De la même manière, l'ordinateur peut générer des nombres au hasard : ce sont des *nombres aléatoires*. La fonction RANDOM (RND) permet à l'ordinateur de les sélectionner. Parfois des chiffres se répètent; mais, dans des séries de nombres aléatoires, la probabilité de ressortir plusieurs fois la même valeur est à peu près nulle.

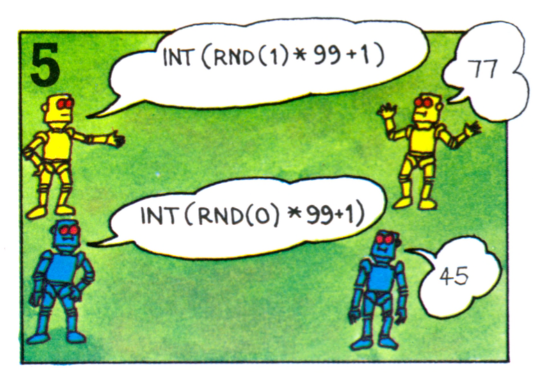


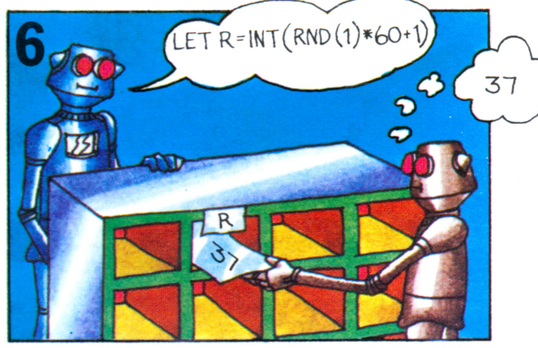




Pour que l'ordinateur génère un nombre aléatoire, on utilise l'instruction RND (RANDOM). Sur certains modèles, il faut ajouter 1 ou 0 entre parenthèses. Consultez votre notice pour confirmation.

L'instruction RND propose toujours un nombre inférieur à 1. Sur certains ordinateurs, on peut écrire un nombre entre parenthèses après RND. On obtient alors une valeur située entre 1 et le nombre placé entre parenthèses.





Sur d'autres ordinateurs, on utilise l'instruction INT (abréviation de "integer" = nombre entier) avant RND [ou RND(1) ou RND(0)]. Il faut ensuite multiplier par le nombre le plus fort que vous souhaitiez et ajouter 1, car vous ne retiendrez que des valeurs supérieures à 1.

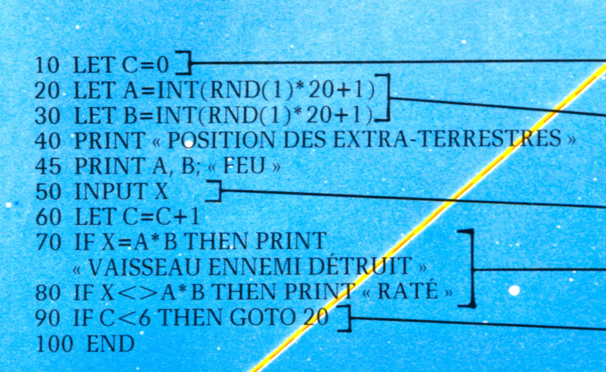
Cette instruction signifie : l'ordinateur tire un nombre aléatoire et le mémorise en R. Les programmes proposés dans ce livre utilisent la formulation suivante : INT(RND(1)*60+1). Il se peut que vous deviez la transformer pour votre ordinateur.



Casse-tête. Pourriez-vous écrire un programme qui permette de tirer un nombre aléatoire situé entre 10 et 20?

L'attaque spatiale

Ce programme de jeu utilise les nombres aléatoires. Dans ce jeu, votre vaisseau spatial est attaqué par une nuée d'extra-terrestres. L'ordinateur du vaisseau repère l'ennemi et vous donne ses positions. Pour détruire chaque extra-terrestre, on doit déterminer l'angle de tir en multipliant les coordonnées qui vous sont communiquées et en tapant le résultat.



```
10 LET C=0
20 LET A=INT(RND(1)*20+1)
30 LET B=INT(RND(1)*20+1)
40 PRINT « POSITION DES EXTRA-TERRESTRES »
45 PRINT A, B; « FEU »
50 INPUT X
60 LET C=C+1
70 IF X=A*B THEN PRINT
   « VAISSEAU ENNEMI DÉTRUIT »
80 IF X<>A*B THEN PRINT « RATÉ »
90 IF C<6 THEN GOTO 20
100 END
```

C permet de compter le nombre de répétitions du programme. A chaque passage, la ligne 60 ajoute 1 à C. (ligne 10)

Ces deux lignes génèrent des nombres aléatoires qui déterminent les positions extra-terrestres. Les coordonnées sont mémorisées en A et B (lignes 20 et 30)

Votre réponse est enregistrée en X. (ligne 50)

Aux lignes 70 et 80, l'ordinateur compare votre réponse à ses propres résultats.

Cette ligne relance le programme si C est inférieur à 6. (ligne 90)

Le programme tourne

L'encadré à droite montre ce qui se passe quand le programme est lancé. Si vous donnez la bonne réponse (le résultat de la multiplication des deux coordonnées), l'ordinateur affichera « vaisseau ennemi détruit ». Si vous vous êtes trompé, vous lirez « raté ».

```
RUN
POSITION DES EXTRA-TERRESTRES
17    3  FEU
?41
RATÉ
POSITION DES EXTRA-TERRESTRES
11    5  FEU
?55
VAISSEAU ENNEMI DÉTRUIT
POSITION DES EXTRA-TERRESTRES
13    6  FEU
```

La ponctuation de la ligne 45 établit ces espaces.

Casse-tête

Sauriez-vous ajouter un autre élément au programme pour comptabiliser le nombre d'ennemis abattus et voir s'afficher votre score? Vous devez introduire une variable, S, lui donner au départ la valeur 0, puis l'augmenter de 1 à chaque réussite.

Dessin aléatoire

```
5 CLS
10 LET X=INT(RND(1)*30+1)
20 LET Y=INT(RND(1)*30+1)
30 PLOT (X,Y)
40 GOTO 10
```

Cette instruction vide l'écran. (ligne 5)

Contrôlez que la valeur donnée aux nombres aléatoires correspond aux capacités de votre ordinateur. (lignes 10 et 20)

Cette instruction permet au programme de se répéter indéfiniment. (ligne 40)

Les instructions CLS, RND, PLOT varient d'un ordinateur à l'autre. Parfois, on devra passer par le mode graphique.



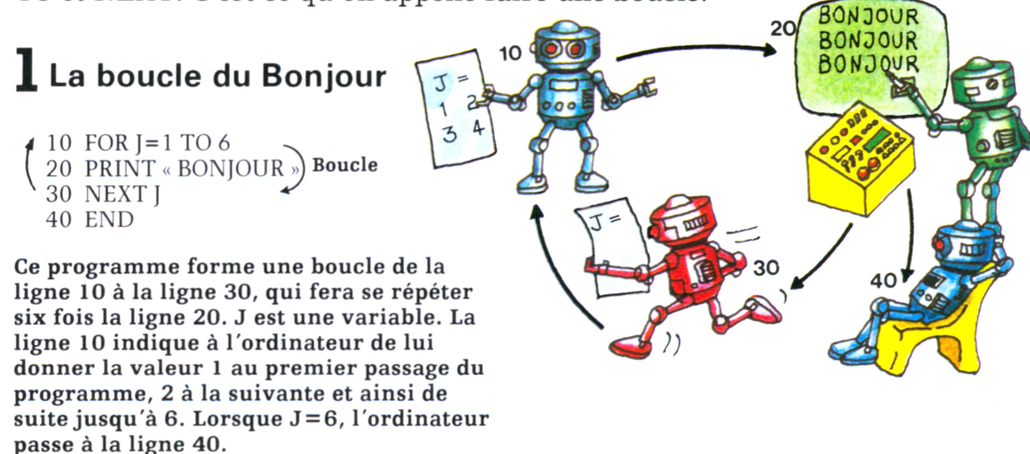
Dans ce programme, on utilise des nombres aléatoires pour afficher des pixels à l'écran. Les lignes 10 et 20 tirent des nombres entre 1 et 30 et les mémorisent en X et Y. La ligne 30 affiche le point dont les coordonnées viennent d'être sorties. Pour arrêter le programme, on doit taper BREAK, ESCAPE, ou le mot qui a cette fonction sur votre ordinateur.

Faisons des boucles

Dans un programme, on a souvent besoin que l'ordinateur répète plusieurs fois la même chose. Vous avez vu, à la page 69, qu'on peut obtenir ce résultat en utilisant GOTO et une variable qui fait fonction de compteur. On peut aussi passer par les instructions FOR, TO et NEXT. C'est ce qu'on appelle faire une boucle.

1 La boucle du Bonjour

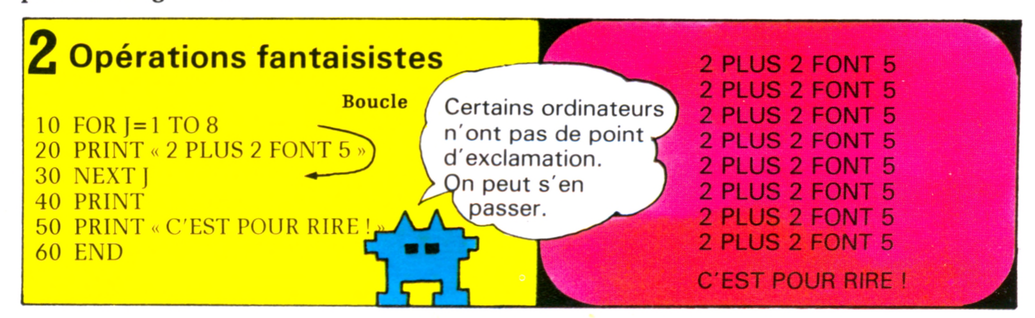
```
10 FOR J=1 TO 6
20 PRINT « BONJOUR »      Boucle
30 NEXT J
40 END
```



Ce programme forme une boucle de la ligne 10 à la ligne 30, qui fera se répéter six fois la ligne 20. J est une variable. La ligne 10 indique à l'ordinateur de lui donner la valeur 1 au premier passage du programme, 2 à la suivante et ainsi de suite jusqu'à 6. Lorsque J=6, l'ordinateur passe à la ligne 40.

2 Opérations fantaisistes

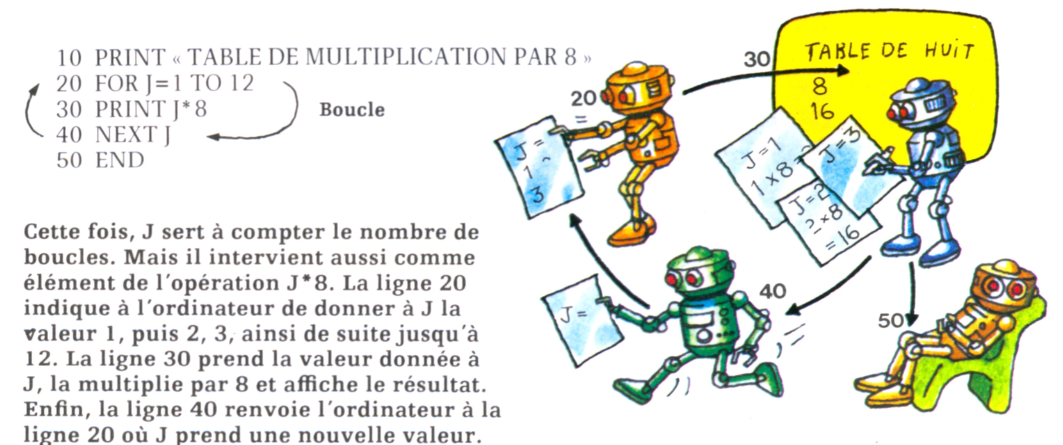
```
10 FOR J=1 TO 8           Boucle
20 PRINT « 2 PLUS 2 FONT 5 »
30 NEXT J
40 PRINT
50 PRINT « C'EST POUR RIRE ! »
60 END
```



Dans ce programme, la boucle des lignes 10 à 30 fait exécuter 8 fois la ligne 20 à l'ordinateur. A chaque fois, il affiche cette même opération fantaisiste. Au bout de huit passages, l'ordinateur continue; la ligne 40 ne sert qu'à lui faire sauter une ligne.

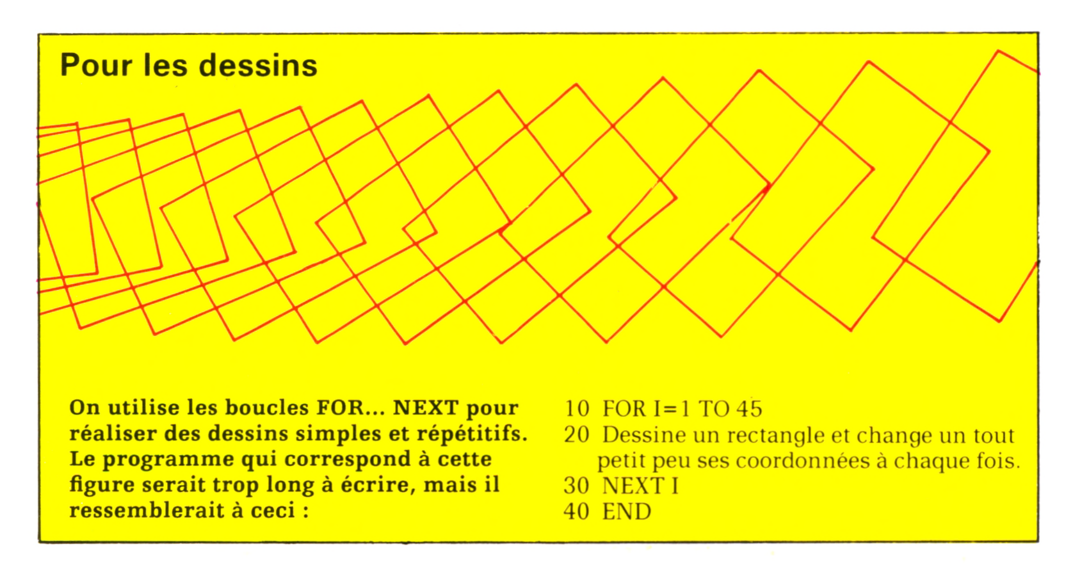
3 Table de multiplication par 8

```
10 PRINT « TABLE DE MULTIPLICATION PAR 8 »
20 FOR J=1 TO 12
30 PRINT J*8              Boucle
40 NEXT J
50 END
```



Cette fois, J sert à compter le nombre de boucles. Mais il intervient aussi comme élément de l'opération J*8. La ligne 20 indique à l'ordinateur de donner à J la valeur 1, puis 2, 3, ainsi de suite jusqu'à 12. La ligne 30 prend la valeur donnée à J, la multiplie par 8 et affiche le résultat. Enfin, la ligne 40 renvoie l'ordinateur à la ligne 20 où J prend une nouvelle valeur.

Pour les dessins



On utilise les boucles FOR... NEXT pour réaliser des dessins simples et répétitifs. Le programme qui correspond à cette figure serait trop long à écrire, mais il ressemblerait à ceci :

```
10 FOR I=1 TO 45
20 Dessine un rectangle et change un tout
   petit peu ses coordonnées à chaque fois.
30 NEXT I
40 END
```

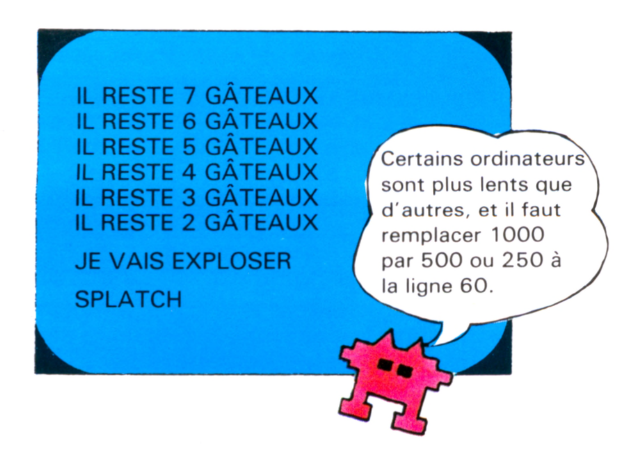
Pas de boucle

Parfois on veut changer la valeur de J, non plus de 1 en 1, mais de 3 en 3 ou de 7 en 7. Pour cela, on utilise l'instruction STEP (pas). Dans le programme ci-dessous, STEP-1 fait décroître de 1 la valeur de J à chaque passage du programme de la ligne 10 à la ligne 40.

L'ordinateur gourmand

Le chiffre 2 arrête le programme après J=2 (quand il ne reste plus qu'un gâteau).

```
5 CLS
10 FOR J=7 TO 2 STEP-1
20 PRINT « IL RESTE »; J; « GÂTEAUX »
30 NEXT J                          Boucle
40 PRINT
50 PRINT « JE VAIS EXPLOSER »
60 FOR K=1 TO 1000
70 REM: NE FAIS RIEN               Boucle
80 NEXT K
90 PRINT
100 PRINT "SPLATCH"
```



Il y a deux boucles dans ce programme. Celle de la ligne 10 à 30 fait s'afficher 6 fois la ligne 20, avec une valeur de J minorée de 1 à chaque fois. Dans la boucle des lignes 60 à 80, l'ordinateur n'a rien à faire; il passe uniquement en revue toutes les valeurs de K comprises entre 1 et 1 000, ce qui provoque une pause dans le déroulement du programme. Les lignes commençant par REM (abréviation de "remark") ne sont pas prises en compte par l'ordinateur; elles ne servent qu'à vous rappeler ce qui se passe dans le programme.

Casse-tête

1. Pouvez-vous transformer le programme de la table de 8 pour que s'affiche "1×8=" et le résultat.

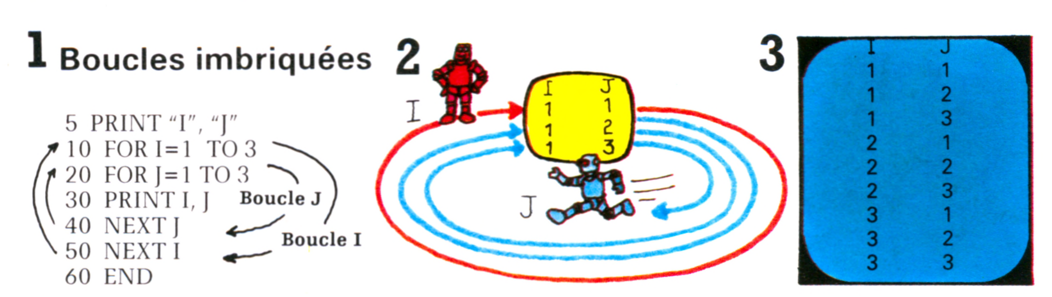
2. Sauriez-vous écrire un programme de la table de N, que vous pourriez utiliser pour n'importe quel nombre que vous entrez sur l'ordinateur? Celui-ci doit tout d'abord vous demander la valeur de N. Puis, au moyen d'une boucle, il calcule et affiche la table de multiplication. Vous pouvez ajouter quelques lignes à la fin du programme pour qu'il vous demande si vous désirez une autre table.

Quelques astuces

Voici d'autres programmes avec des boucles. On y découvre comment utiliser des boucles à l'intérieur d'autres boucles pour répéter plusieurs données à des rythmes différents. Ce sont des *boucles imbriquées.*

1 Boucles imbriquées

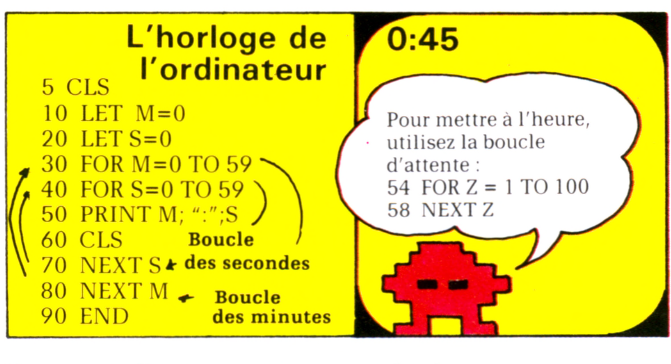
```
5 PRINT "I", "J"
10 FOR I=1 TO 3
20 FOR J=1 TO 3
30 PRINT I, J
40 NEXT J
50 NEXT I
60 END
```



Le programme comporte une boucle I et une boucle J. La boucle J est imbriquée dans la boucle I. Pour chaque passage de la boucle I, la boucle J se répète trois fois, affichant à chaque tour une nouvelle valeur de J. Vous voyez dans l'encadré ci-dessus les résultats de ce programme. Les espacements sont dus aux virgules.

L'horloge de l'ordinateur

```
5 CLS
10 LET M=0
20 LET S=0
30 FOR M=0 TO 59
40 FOR S=0 TO 59
50 PRINT M; ":";S
60 CLS
70 NEXT S
80 NEXT M
90 END
```



Des bugs dans les boucles

```
10 FOR I = 1 TO 4
20 FOR J = 1 TO 4
30 PRINT I
40 PRINT J
50 NEXT I
60 NEXT J
```

Les deux parties d'une boucle imbriquée doivent se trouver à l'intérieur de l'autre boucle.

Dans un ordinateur se trouve une horloge électronique qui imprime la fréquence de son travail. L'horloge émet entre 1 et 4 millions d'impulsions à la seconde. Ce programme indique à l'ordinateur de se comporter comme une horloge numérique. On utilise des boucles imbriquées, l'une pour compter les secondes, l'autre pour les minutes. La boucle imbriquée tourne 59 fois pour un seul passage de la boucle des minutes. Quand vous testerez ce programme sur votre appareil, il se peut qu'il tourne trop vite. Ajoutez alors une boucle d'attente et adaptez-la en changeant le chiffre de telle sorte que votre ordinateur avance au même rythme qu'une véritable horloge.

Test des nombres aléatoires

```
10 FOR I=1 TO 1000
20 LET R=INT(RND(1)*6+1)
30 IF R=1 THEN LET A=A+1
40 IF R=2 THEN LET B=B+1
50 IF R=3 THEN LET C=C+1
60 IF R=4 THEN LET D=D+1
70 IF R=5 THEN LET E=E+1
80 IF R=6 THEN LET F=F+1
90 NEXT I
100 PRINT
   « TERMINE »
110 PRINT A, B, C
120 PRINT D, E, F
130 END
```

Le déroulement de ce programme est très, très long. Vous pouvez le raccourcir en remplaçant à la ligne 10 le nombre 1 000 par 500 ou même 250.

```
RUN
TERMINÉ
162        168        167
160        187        156
```

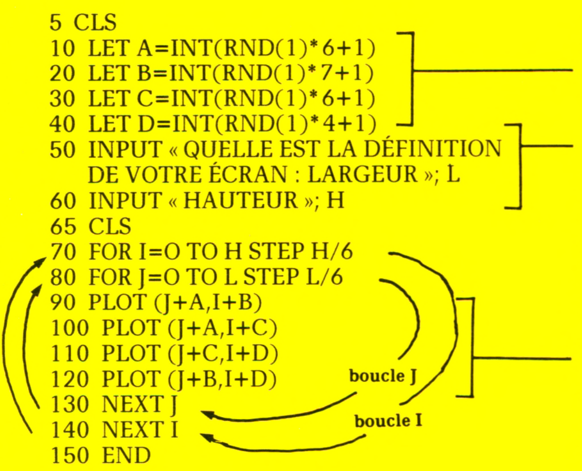
Ce programme vérifie le fonctionnement de RANDOM. La boucle de la ligne 10 à 90 fait tirer à l'ordinateur un chiffre entre 1 et 6 un millier de fois. L'ordinateur totalise combien de fois chaque nombre a été mémorisé en variable A, B, C, D, E ou F, puis affiche les résultats*.

* Sur certains ordinateurs, comme le SINCLAIR ZX81 ou le BBC, il faut écrire quelques lignes en début de programme pour initialiser les variables.

Répétition de dessins

Ce programme utilise des boucles imbriquées pour répéter un même petit dessin partout sur l'écran. Il semble ardu, mais si on le regarde soigneusement, si l'on découvre l'utilité de chaque ligne, on comprendra rapidement son mode de fonctionnement. C'est le tirage qui décidera de la forme du dessin; il sera donc différent à chaque fois que vous lancerez le programme.

Pour les ordinateurs à haute résolution graphique, prenez des nombres aléatoires plus importants. Ainsi, pour le BBC, changez les chiffres des lignes 10 à 40.



```
5 CLS
10 LET A=INT(RND(1)*6+1)
20 LET B=INT(RND(1)*7+1)
30 LET C=INT(RND(1)*6+1)
40 LET D=INT(RND(1)*4+1)
50 INPUT « QUELLE EST LA DÉFINITION
   DE VOTRE ÉCRAN : LARGEUR »; L
60 INPUT « HAUTEUR »; H
65 CLS
70 FOR I=O TO H STEP H/6
80 FOR J=O TO L STEP L/6
90 PLOT (J+A,I+B)
100 PLOT (J+A,I+C)
110 PLOT (J+C,I+D)
120 PLOT (J+B,I+D)
130 NEXT J
140 NEXT I
150 END
```


Ces lignes tirent les nombres aléatoires et les mémorisent en A, B, C ou D.

Les lignes 50 et 60 demandent la largeur (L) et la hauteur (H) de votre écran.

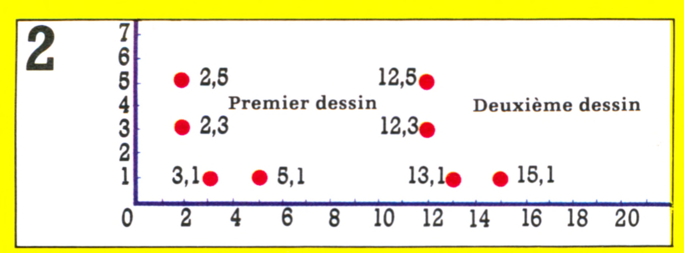
La boucle I compte le nombre de fois où le dessin se répète à l'écran. A chaque fois s'ajoute à I la hauteur de l'écran divisé par 6, si bien que le dessin s'affiche 6 fois en hauteur sur l'écran.

A chaque fois que les boucles se répètent, les lignes 90 à 120 font tracer à l'ordinateur 4 points correspondant aux valeurs de I et de J augmentées du nombre aléatoire.

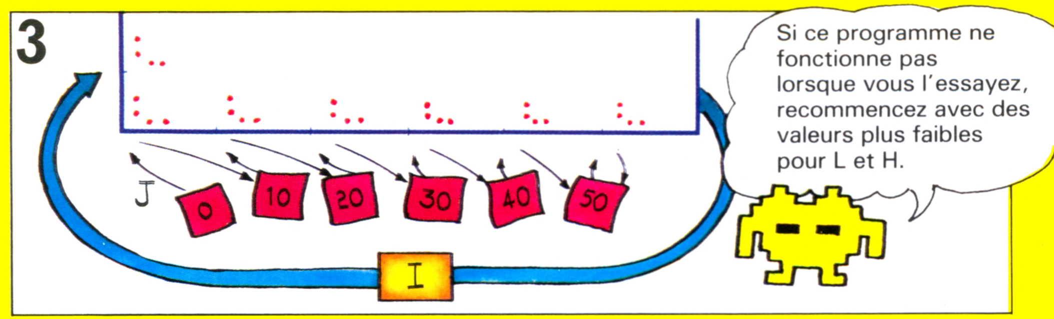
La boucle J compte le nombre d'apparitions du dessin sur la largeur de l'écran. Elle fonctionne comme la boucle I.



Imaginons que l'ordinateur ait tiré les nombres 2, 5, 3 et 1 et que la largeur et la hauteur de l'écran soient toutes deux de 60.



Au premier tour, I et J valent 0; l'ordinateur trace donc le premier dessin uniquement en fonction du nombre tiré. La ligne 130 le renvoie chercher la valeur suivante de J, qui est J+60/6, c'est-à-dire 10. Il trace alors le second dessin en utilisant le nombre aléatoire, plus 10 pour J. Le dessin se répète sur l'écran.



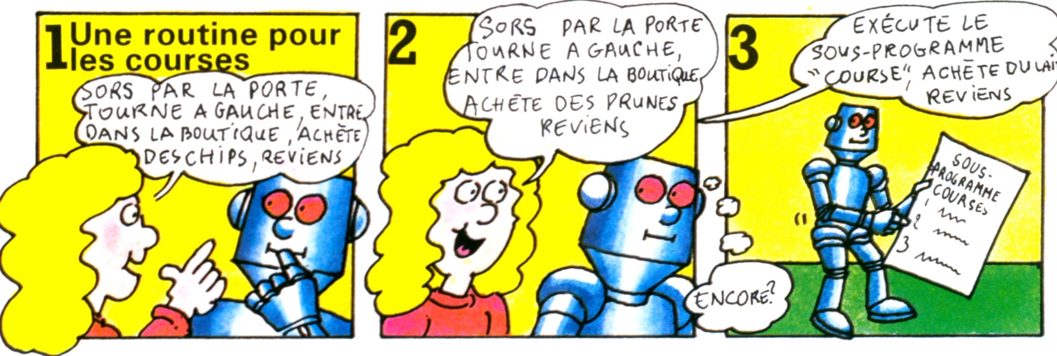
L'ordinateur répète 6 fois la boucle J, en ajoutant à chaque fois 10 à la valeur de J. Il revient ensuite prendre la valeur suivante de I qui est 10. J est de nouveau égal à 0 et l'ordinateur trace une nouvelle ligne de dessins en prenant 10 comme valeur de I et en augmentant à nouveau J de 10 en 10.



Casse-tête. Inventez un programme dans lequel la forme qui se répète serait un envahisseur extra-terrestre. **Vous trouverez en page 45 quelques trucs pour vous aider.**

Sous-programmes

Un sous-programme est une petite routine qui se trouve à l'intérieur d'un autre programme. Sa fonction est d'exécuter des tâches particulières comme additionner des nombres ou tenir un score. On peut y envoyer l'ordinateur dès que l'on veut que cette tâche soit effectuée, ce qui évite d'écrire plusieurs fois les mêmes lignes et rend le programme plus facile à lire et à entrer dans l'appareil.



Imaginez que vous ayez un robot pour vous aider, et que vous puissiez programmer ses déplacements. Si vous voulez qu'il aille acheter quelque chose dans une boutique, il faut lui donner des indications précises sur la façon de s'y rendre. A chaque fois que vous voudrez qu'il fasse des courses, vous devrez lui répéter les mêmes ordres. Il serait beaucoup plus facile de programmer une routine de courses et de lui dire de s'y référer.

4 Programme pour faire ses courses

```
10 PRINT « DE QUOI AVEZ-VOUS
          BESOIN AU MAGASIN? »
20 INPUT X$
30 GOSUB 100
40 PRINT « AUTRE CHOSE? »
50 INPUT M$
60 IF M$=« OUI » THEN GOTO 10
70 STOP

100 REM : ROUTINE COURSES
110 PRINT « SORS, TOURNE À GAUCHE »
120 PRINT « ENCORE À GAUCHE,
           ENTRE DANS LA BOUTIQUE »
130 PRINT « ACHÈTE »; X$; « REVIENS »
140 RETURN
```

La ligne 30 envoie l'ordinateur à la première ligne de la routine.

STOP, à la fin du programme principal, empêche l'ordinateur d'enchaîner sur la routine.

Étiqueter une routine avec REM permet de savoir à quoi elle sert.

L'ordinateur est renvoyé à la ligne 40, juste après GOSUB.

Si vous oubliez la ligne RETURN, c'est un bug.

En BASIC, pour dire à l'ordinateur de se rendre à un sous-programme, on utilise les instructions GOSUB et RETURN. GOSUB doit être suivi du numéro de la première ligne d'instructions du sous-programme. RETURN, placé à la fin de la routine, n'a pas besoin d'être suivi d'un numéro : il renvoie automatiquement à la ligne d'instructions située en dessous de celle qui a fait quitter le programme principal. On peut envoyer l'ordinateur à une routine autant de fois que l'on veut et d'où l'on veut dans le programme.

Des programmes avec gosub

On utilise des sous-programmes chaque fois que l'on veut exécuter plusieurs fois une même tâche. Voici quelques programmes recourant à des routines.

Programme numérique

```
50 INPUT A
60 INPUT B
70 GOSUB 250
80 PRINT « A DIVISÉ PAR B = »; A/B
90 GOTO 50
250 REM: SOUS-PROGRAMME D'ARRÊT
260 IF A=0 AND B=0 THEN STOP
70 RETURN
```

Ce sous-programme permet de sortir du programme principal. Si vous voulez arrêter de diviser, entrez 0 en INPUT aux lignes 50 et 60. On ne met pas de STOP dans ce programme, puisque la ligne 90 renvoie à la ligne 50.

Programme de conversion

```
100 INPUT « DISTANCE »; M
110 INPUT « TEMPS »; T
120 GOSUB 200
130 PRINT « LA VITESSE MOYENNE ÉTAIT DE »
140 PRINT M/T; "MPH ET"; K/M; "KMH"
150 STOP
200 REM: SOUS-PROGRAMME DE CONVERSION
DES MILES
210 LET KM=M*1609
220 RETURN
```

Voici un sous-programme dont le rôle est de convertir les miles en kilomètres. Il peut servir dans de nombreux programmes. Vérifiez seulement le nom des variables.

Programme de tracé de cercles

```
1 Centre du cercle = X, Y
2 Rayon du cercle = R
3 Couleur = X
4 GOSUB 10
5 GOTO 1
10 REM: sous-programme pour tracer
des cercles
11 Tracé d'un cercle de centre X, Y,
de rayon R et de couleur
12 RETURN
```

Ce programme est écrit en français, non en BASIC. Il vous en donne l'idée générale.

On utilise les sous-programmes dans les applications graphiques : ils permettent de tracer des diagrammes à partir des nombres trouvés dans le programme principal. Dans ce programme, on pourrait dessiner de nombreux cercles différents, simplement en entrant dans l'ordinateur des informations aux lignes 1 à 5.

Programme de QUIZ

```
5 LET C=0
10 PRINT « QUAND CELA A-T-IL ÉTÉ INVENTÉ? »
20 READ C$, F
30 PRINT C$
40 INPUT A
50 LET C=C+1
60 IF C=3 THEN STOP
70 GOSUB 100
80 GOTO 10
100 REM: SOUS-PROGRAMME DE RÉPONSES
110 IF ABS(A-F)10<THEN PRINT « OUI »
120 IF ABS(A-F)>10 THEN PRINT « NON »
130 PRINT « NOUVEL ESSAI »
140 RETURN
200 DATA TÉLÉPHONE, 1876, IMPRIMERIE, 1450, BICYCLETTE, 1791
```

A la ligne 20, l'ordinateur se rend à la ligne DATA et prend le premier article pour le placer en C$ et F.

Le mot mémorisé en C$ s'affiche ici.

Le compteur C arrête le programme une fois qu'il s'est déroulé 3 fois, car il n'y a que 3 articles en C$ et F.

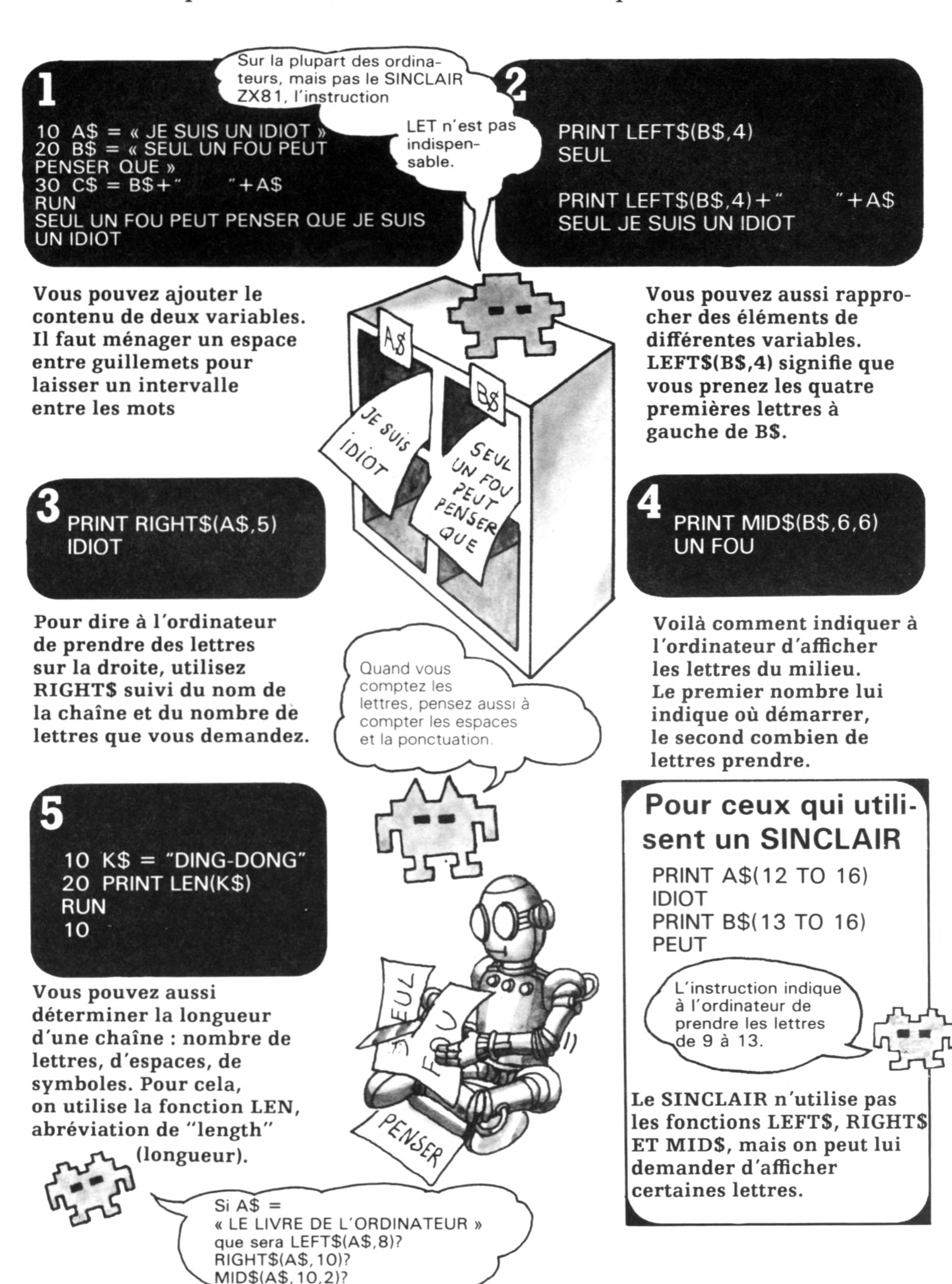
Voilà le sous-programme.

A chaque nouveau passage du programme, les données en C$ et F sont remplacées par les suivantes.

Ce programme a recours à une routine pour contrôler les réponses. Les bonnes réponses sont mémorisées en F et les propositions du joueur en A. Aux lignes 100 et 110 de la routine, l'ordinateur compare A et F. ABS veut dire valeur absolue : l'appareil compare A et F sans tenir compte des signes négatifs. Si la différence est inférieure à 10, il affiche « OUI », autrement « NON ».

Jouons avec les mots

La plupart des ordinateurs savent analyser les mots mémorisés en variable. Cette analyse les rend capables de diverses applications : ils peuvent détailler le contenu d'une variable et déterminer si un mot ou une lettre particulière s'y trouve, ce qui permet de vérifier les mots introduits par un utilisateur.
Ils savent aussi réorganiser les lettres et les mots en un ordre différent et combiner plusieurs variables. Voici comment procéder en BASIC.



1

```
10 A$ = « JE SUIS UN IDIOT »
20 B$ = « SEUL UN FOU PEUT
PENSER QUE »
30 C$ = B$+"      "+A$
RUN
SEUL UN FOU PEUT PENSER QUE JE SUIS
UN IDIOT
```

Vous pouvez ajouter le contenu de deux variables. Il faut ménager un espace entre guillemets pour laisser un intervalle entre les mots

2

```
PRINT LEFT$(B$,4)
SEUL

PRINT LEFT$(B$,4)+"      "+A$
SEUL JE SUIS UN IDIOT
```

Vous pouvez aussi rapprocher des éléments de différentes variables. LEFT$(B$,4) signifie que vous prenez les quatre premières lettres à gauche de B$.

3

```
PRINT RIGHT$(A$,5)
IDIOT
```

Pour dire à l'ordinateur de prendre des lettres sur la droite, utilisez RIGHT$ suivi du nom de la chaîne et du nombre de lettres que vous demandez.

4

```
PRINT MID$(B$,6,6)
UN FOU
```

Voilà comment indiquer à l'ordinateur d'afficher les lettres du milieu. Le premier nombre lui indique où démarrer, le second combien de lettres prendre.

5

```
10 K$ = "DING-DONG"
20 PRINT LEN(K$)
RUN
10
```

Vous pouvez aussi déterminer la longueur d'une chaîne : nombre de lettres, d'espaces, de symboles. Pour cela, on utilise la fonction LEN, abréviation de "length" (longueur).

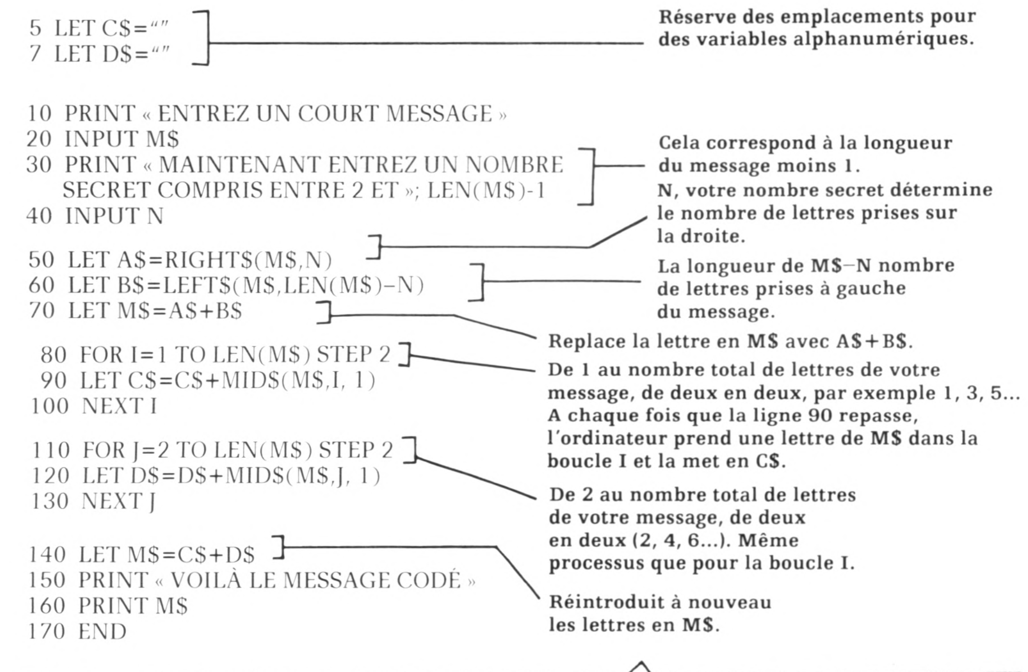
Pour ceux qui utilisent un SINCLAIR

```
PRINT A$(12 TO 16)
IDIOT
PRINT B$(13 TO 16)
PEUT
```

Le SINCLAIR n'utilise pas les fonctions LEFT$, RIGHT$ ET MID$, mais on peut lui demander d'afficher certaines lettres.

Codons des messages

Ce programme code les mots automatiquement. Ce sont des programmes similaires, mais bien plus compliqués, qu'utilisent les services secrets pour écrire et décoder des messages.
Pour comprendre le fonctionnement de ce programme, écrivez un message sur une feuille de papier, entrez le programme sur l'ordinateur. Vous vous rendrez compte, au fur et à mesure, des transformations apportées à votre message initial.



```
5 LET C$=""
7 LET D$=""

10 PRINT « ENTREZ UN COURT MESSAGE »
20 INPUT M$
30 PRINT « MAINTENANT ENTREZ UN NOMBRE
   SECRET COMPRIS ENTRE 2 ET »; LEN(M$)-1
40 INPUT N

50 LET A$=RIGHT$(M$,N)
60 LET B$=LEFT$(M$,LEN(M$)-N)
70 LET M$=A$+B$

80 FOR I=1 TO LEN(M$) STEP 2
90 LET C$=C$+MID$(M$,I, 1)
100 NEXT I

110 FOR J=2 TO LEN(M$) STEP 2
120 LET D$=D$+MID$(M$,J, 1)
130 NEXT J

140 LET M$=C$+D$
150 PRINT « VOILÀ LE MESSAGE CODÉ »
160 PRINT M$
170 END
```

Réserve des emplacements pour des variables alphanumériques. (lignes 5–7)

Cela correspond à la longueur du message moins 1. (ligne 30)

N, votre nombre secret détermine le nombre de lettres prises sur la droite. (ligne 50)

La longueur de M$–N nombre de lettres prises à gauche du message. (ligne 60)

Replace la lettre en M$ avec A$+B$. (ligne 70)

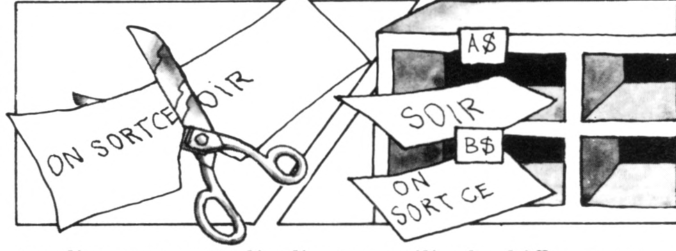
De 1 au nombre total de lettres de votre message, de deux en deux, par exemple 1, 3, 5... A chaque fois que la ligne 90 repasse, l'ordinateur prend une lettre de M$ dans la boucle I et la met en C$. (ligne 80)

De 2 au nombre total de lettres de votre message, de deux en deux (2, 4, 6...). Même processus que pour la boucle I. (ligne 110)

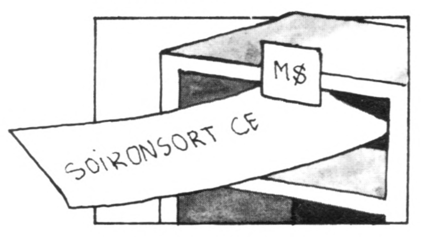
Réintroduit à nouveau les lettres en M$. (ligne 140)



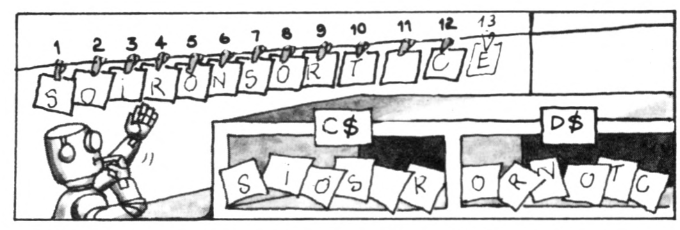
Imaginons que le message soit « On sort ce soir » et le chiffre secret 4. Ces données sont mémorisées en M$ et N.



Aux lignes 50 et 60, l'ordinateur utilise le chiffre secret pour segmenter le message. A la ligne 50, il prend 4 lettres à la droite du message et les met en A$. A la ligne 60, il met le reste du message en B$.



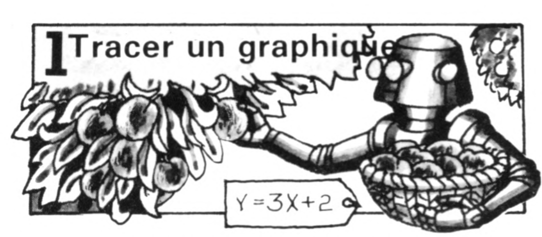
A la ligne 70, il réunit A$ et B$. Ainsi les dernières lettres du message se retrouvent-elles à l'avant.



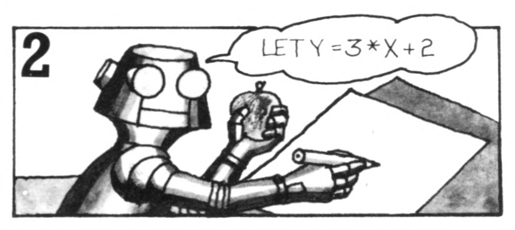
Chaque fois que la boucle I se répète, elle place une lettre différente en C$ (comme S, I, O). De même, à chaque fois que la boucle J se répète, elle place une lettre en D$ (comme O, R, N). Puis l'ordinateur réunit C$ et D$ pour établir le message codé.

Graphiques et symboles

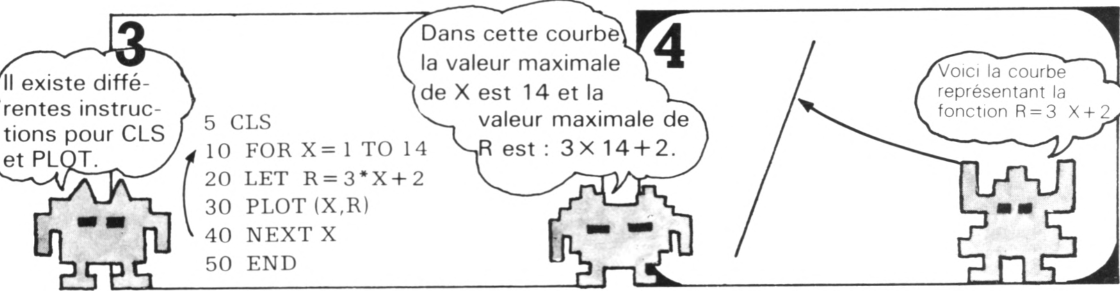
Vous pouvez programmer un ordinateur pour qu'il donne les informations de différentes façons : mots, nombres, dessins, graphiques. On facilite ainsi la compréhension de sujets complexes en les illustrant par des graphiques, des images ou des symboles.



Prenons un pêcher dont le rendement en fruits croît chaque année en fonction de son âge. On peut poser ce fait en équation, par exemple R=3X+2 (R représente le rendement et X l'âge). Telle quelle, la notion est difficile à saisir. Un graphique sera le bienvenu.



Avec un ordinateur, il est très facile de tracer la courbe de déplacement de R en fonction de X. Pour tracer cette courbe, déterminer la valeur de R pour chaque valeur de X. Vous obtiendrez ce résultat en programmant l'instruction LET R=3*X+2.



Voilà le programme qui permet de tracer cette courbe. La boucle donne successivement à X toutes les valeurs comprises entre 1 et 14; à chaque fois, la ligne 20 en déduit la valeur correspondante de R. La ligne 30 affiche le point X, R à l'écran. Dans ce type de programme, on doit vérifier que les valeurs maximales de X et R sont contenues dans l'écran. Autrement, c'est un bug.

Ordinateurs et mathématiques

Dans les calculs multiples, comme 3*X+2, l'ordinateur effectue toujours les multiplications et les divisions avant d'opérer les additions et les soustractions. Il donnera donc le même résultat pour les deux opérations suivantes :

```
PRINT 4*6+8        PRINT 8+4*6
32                 32
```

Si vous voulez modifier l'ordre des opérations, utilisez des parenthèses :

```
PRINT (8+4)*6
72
```

L'ordinateur a d'abord additionné 8 et 4, puis il a multiplié cette somme par 6.

Casse-tête

PENSEZ À UN NOMBRE
MULTIPLIEZ-LE PAR 2, AJOUTEZ 4
DIVISEZ PAR 2, AJOUTEZ 7
MULTIPLIEZ PAR 8, ENLEVEZ 12
DIVISEZ PAR 4, ENLEVEZ 11
DITES-MOI COMBIEN IL RESTE.
LE NOMBRE AUQUEL VOUS AVIEZ PENSÉ ÉTAIT :

Sauriez-vous écrire un programme qui traite ce jeu bien connu? (Pour retrouver le premier chiffre, vous enlevez 4 au « reste », puis vous divisez par 2.)

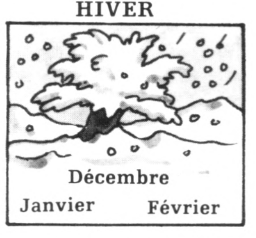
Programme d'anniversaire

Voici une autre méthode pour afficher des informations à l'écran. On utilise des signes pour comparer le nombre de personnes nées en chaque saison de l'année. Un programme de ce type peut servir, par exemple, à comparer les apparitions d'une espèce d'oiseaux selon les saisons, ou le nombre de buts marqués par une équipe de football. Avant de se lancer dans la rédaction d'une liste aussi longue, il est bon d'établir un plan de programme.

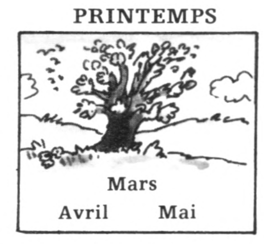
Plan du programme

But : comparer le nombre des personnes dont l'anniversaire tombe en hiver, printemps, été, automne.

HIVER



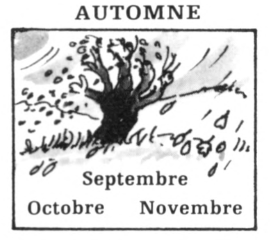
PRINTEMPS



ÉTÉ



AUTOMNE

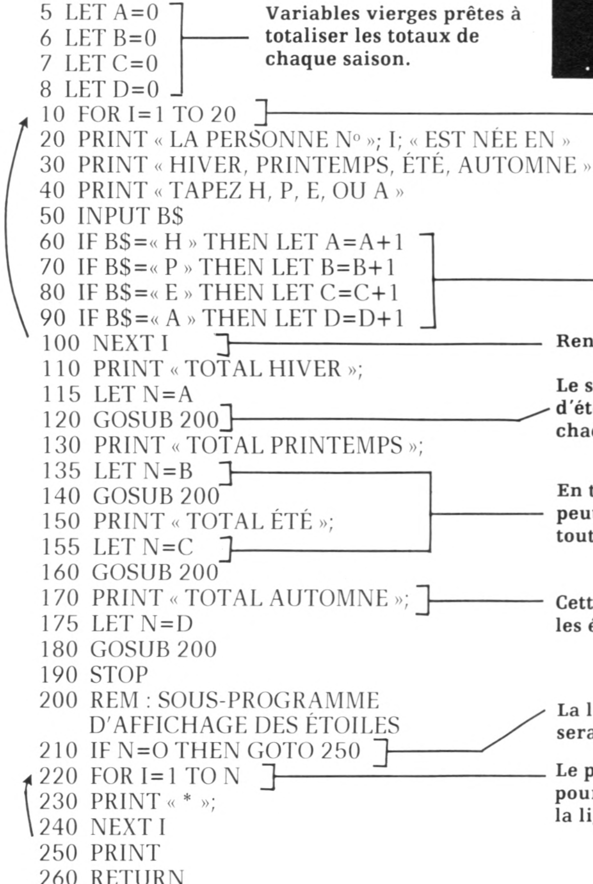


1. **Entrer les données dans l'ordinateur (saison de la naissance) pour un échantillon de 20 personnes.**
2. **Mémoriser les données.**
3. **Afficher les données à l'écran.**

Exemple de fonctionnement



Le programme



```
5 LET A=0
6 LET B=0
7 LET C=0
8 LET D=0
10 FOR I=1 TO 20
20 PRINT « LA PERSONNE Nº »; I; « EST NÉE EN »
30 PRINT « HIVER, PRINTEMPS, ÉTÉ, AUTOMNE »
40 PRINT « TAPEZ H, P, E, OU A »
50 INPUT B$
60 IF B$=« H » THEN LET A=A+1
70 IF B$=« P » THEN LET B=B+1
80 IF B$=« E » THEN LET C=C+1
90 IF B$=« A » THEN LET D=D+1
100 NEXT I
110 PRINT « TOTAL HIVER »;
115 LET N=A
120 GOSUB 200
130 PRINT « TOTAL PRINTEMPS »;
135 LET N=B
140 GOSUB 200
150 PRINT « TOTAL ÉTÉ »;
155 LET N=C
160 GOSUB 200
170 PRINT « TOTAL AUTOMNE »;
175 LET N=D
180 GOSUB 200
190 STOP
200 REM : SOUS-PROGRAMME
    D'AFFICHAGE DES ÉTOILES
210 IF N=O THEN GOTO 250
220 FOR I=1 TO N
230 PRINT « * »;
240 NEXT I
250 PRINT
260 RETURN
```

Variables vierges prêtes à totaliser les totaux de chaque saison. (lignes 5 à 8)

Cette boucle oblige l'ordinateur à poser la question pour chaque personne de l'échantillon. (ligne 10)

De la ligne 60 à la ligne 100, l'ordinateur compare la réponse en B$ avec les données et ajoute 1 à la variable correspondante.

Renvoie l'ordinateur à la ligne 10. (ligne 100)

Le sous-programme permet d'afficher un nombre d'étoiles correspondant à la valeur de chacune des variables. (ligne 120)

En transposant le total en N, le programme peut se servir de la même routine pour toutes les saisons. (lignes 135 à 155)

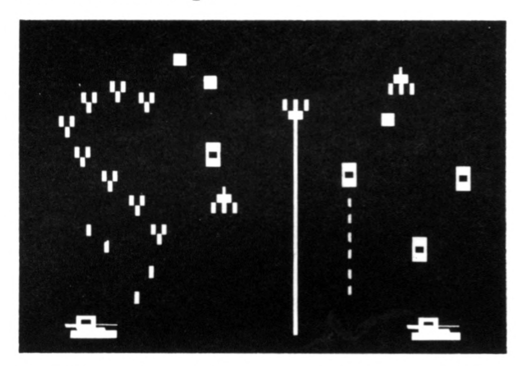
Cette ligne oblige l'ordinateur à afficher les étoiles sur une même ligne. (ligne 170)

La ligne 210 est utile au cas où personne ne serait né en une quelconque saison.

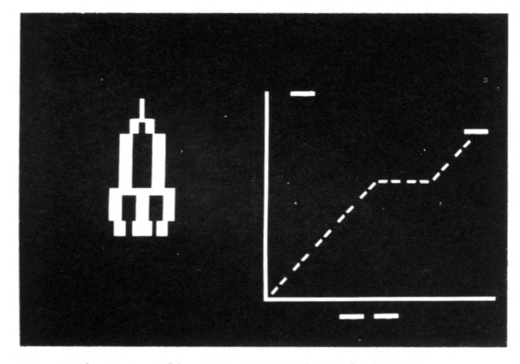
Le programme principal utilise N comme total pour A, B, C ou D. L'ordinateur reproduit la ligne 230 « N » fois. (ligne 220)

Encore des graphiques

Voici deux pages qui montrent comment utiliser PLOT et UNPLOT pour créer des dessins sur l'écran. Ces graphiques animés servent dans les programmes de jeu ou pour illustrer des trajectoires, des principes de gravité ou de balistique.



Dans les jeux vidéo à usage personnel ou public, les dessins sont contrôlés par un petit ordinateur programmé uniquement pour ces jeux. Les instructions sont délivrées en langage machine et non en BASIC.



Un micro-ordinateur multi-tâches programmé en BASIC ne peut réaliser que des dessins plus lents et plus simples. Il n'est pas capable en effet de traiter assez rapidement toutes les instructions concernant l'écran.

1 Plot/unplot

```
10 LET X=1
20 LET Y=1
30 PLOT (X,Y)
40 UNPLOT (X,Y)
50 LET X=X+1
60 LET Y=Y+1
70 GOTO 30
```



Par ce court programme, un point lumineux se déplace à travers l'écran. N'oubliez pas que les instructions PLOT et UNPLOT varient selon les ordinateurs.

2



Lorsque le point atteint le bord de l'écran, le programme s'arrête : les valeurs de X et Y se trouvent alors au-delà des limites de l'écran de l'ordinateur.

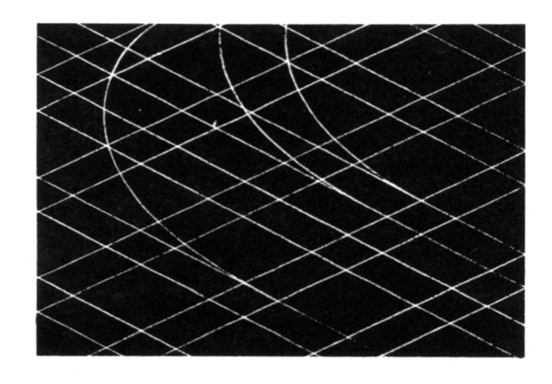
3

Les jeux de balles vidéo utilisent des programmes du type de celui que nous venons de voir. Il existe des règles simples à appliquer pour que le jeu se poursuive lorsque la balle touche les bords de l'écran.

Lorsque la balle touche le haut de l'écran, la valeur qui aurait dû être ajoutée à Y se trouve au contraire retranchée. De même, lorsqu'elle atteint le bord droit de l'écran, la valeur de X est minorée.

Tracé de ligne

Ce programme permet de tracer une ligne continue sur l'écran. Lorsqu'elle se heurte à un bord de l'écran, le programme la renvoie dans une autre direction. Comme on n'utilise pas l'instruction UNPLOT, on obtient un dessin. Vous voyez sur la figure de droite ce qui se passe quand vous lancez le programme. A la ligne 100, l'ordinateur est programmé pour tracer jusqu'à 10 000 pixels. Vous pouvez changer ce chiffre pour raccourcir le programme, ou l'arrêter en cours d'exécution lorsque le dessin vous plaît.



```
 10 REM: PASSER EN MODE GRAPHIQUE SI NÉCESSAIRE
 20 PRINT "DÉFINITION DE L'ÉCRAN,
LARGEUR";
 30 INPUT L
 40 PRINT « HAUTEUR »;
 50 INPUT H

 55 CLS
 60 LET X=L/2
 70 LET Y=H/2
 80 LET S=1
 90 LET T=1
100 FOR I=1 TO 10000
110 LET S=S+(INT(RND(1)*10+1)-5)/50
120 LET X=X+S
130 LET Y=Y+T

140 IF X<5 THEN LET S=-S
150 IF X>L-5 THEN LET S=-S
160 IF Y<5 THEN LET T=-T
170 IF Y>H-5 THEN LET T=-T

180 GOSUB 300
190 NEXT I
200 STOP
300 REM: PLOT LINE
310 PLOT (X,Y)
320 RETURN
```

Les lignes 20 à 50 vous demandent le nombre de points de votre écran. Les points-virgules permettent à la réponse de s'afficher à la même ligne que la question.

Ainsi X et Y démarrent du centre de l'écran. (lignes 60-70)

S et T sont les valeurs qui seront ajoutées à X et Y pour les faire se déplacer. (lignes 80-90)

La boucle des lignes 100 à 190 se répète 10 000 fois. A chaque passage, X et Y changent légèrement de valeur.

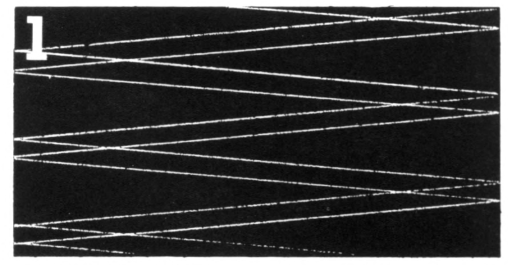
Cette instruction donne une très petite valeur à ajouter à X. (ligne 110)

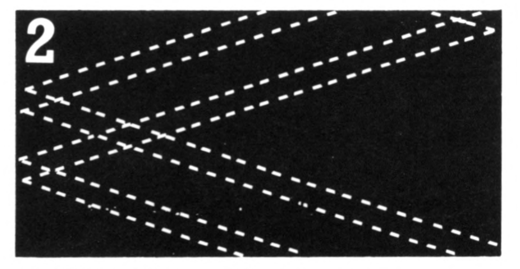
Ces lignes contrôlent les bords de l'écran et inversent S et T lorsque X et Y arrivent à moins de cinq points du bord. (lignes 140-170)

Envoie l'ordinateur au sous-programme qui affiche le point. (ligne 180)

Trace le pixel correspondant aux valeurs données de X et Y. (ligne 310)

Essais

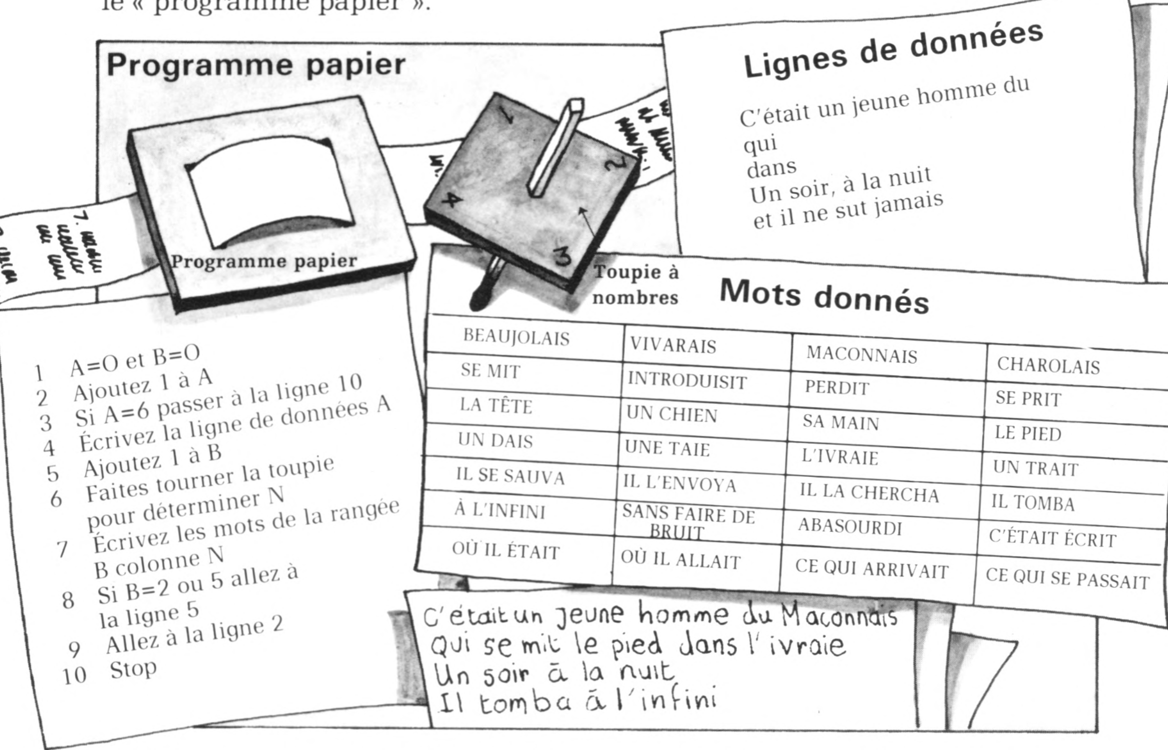




La ligne 110 ajoute un très petit nombre aléatoire à X et à chaque passage de boucle, ce qui fait osciller la ligne à travers l'écran. Si vous avez un ordinateur, amusez-vous à effacer cette instruction : les lignes deviendront parallèles. Essayez de changer les nombres aux lignes 80 et 90, disons 5 ou 10 (ou plus sur un ordinateur à très haute résolution). Vous obtiendrez des lignes pointillées.

Programmons des poèmes drôles

Les quelques pages suivantes vous indiquent comment écrire un programme capable de composer des dizaines de poésies. Vous en aviez déjà trouvé une version dans l'*Introduction aux ordinateurs.* Vous y appreniez comment établir sur le papier un programme qui tire au sort les diverses rimes : c'est le « programme papier ».



BEAUJOLAIS	VIVARAIS	MACONNAIS	CHAROLAIS
SE MIT	INTRODUISIT	PERDIT	SE PRIT
LA TÊTE	UN CHIEN	SA MAIN	LE PIED
UN DAIS	UNE TAIE	L'IVRAIE	UN TRAIT
IL SE SAUVA	IL L'ENVOYA	IL LA CHERCHA	IL TOMBA
À L'INFINI	SANS FAIRE DE BRUIT	ABASOURDI	C'ÉTAIT ÉCRIT
OÙ IL ÉTAIT	OÙ IL ALLAIT	CE QUI ARRIVAIT	CE QUI SE PASSAIT

Voilà le programme papier. Il ressemble à du BASIC, mais ne fonctionnerait pas sur un véritable ordinateur. Mots et phrases du poème sont conservés sur des morceaux de papier et le programme indique lequel sélectionner. La toupie donne un nombre aléatoire entre 1 et 4.

1 Traduction du programme en BASIC

Ce programme ne peut pas encore fonctionner sur un ordinateur.

```
10 LET A=O
20 LET B=O
30 LET A=A+1
40 IF A=6 THEN STOP
50 Écrire la ligne de données A
60 LET B=B+1
70 LET N=INT(RND(1)*4+1)
80 Écrire les données
   de la rangée B colonne N
90 IF B=2 THEN GOTO 60
100 IF B=5 THEN GOTO 60
110 GOTO 30
120 END
```

Ces lignes libèrent des variables. (10–20)

Les lignes 30 et 40 tiennent compte du nombre de données que l'ordinateur a sélectionnées.

Les lignes 50 et 80 ne sont pas en BASIC.

La ligne 60 compte le nombre de mots en données.

Détermine un nombre aléatoire entre 1 et 4. (70)

Les lignes 90 et 100 envoient l'ordinateur sélectionner une autre ligne de données.

La plus grande partie du programme se traduit sans difficulté en BASIC. Toutefois les lignes 50 et 80 posent problème. Il faut donner à l'ordinateur une méthode de stockage et de tirage au sort des lignes et des mots qui lui permette de les sélectionner au bon moment dans le poème.

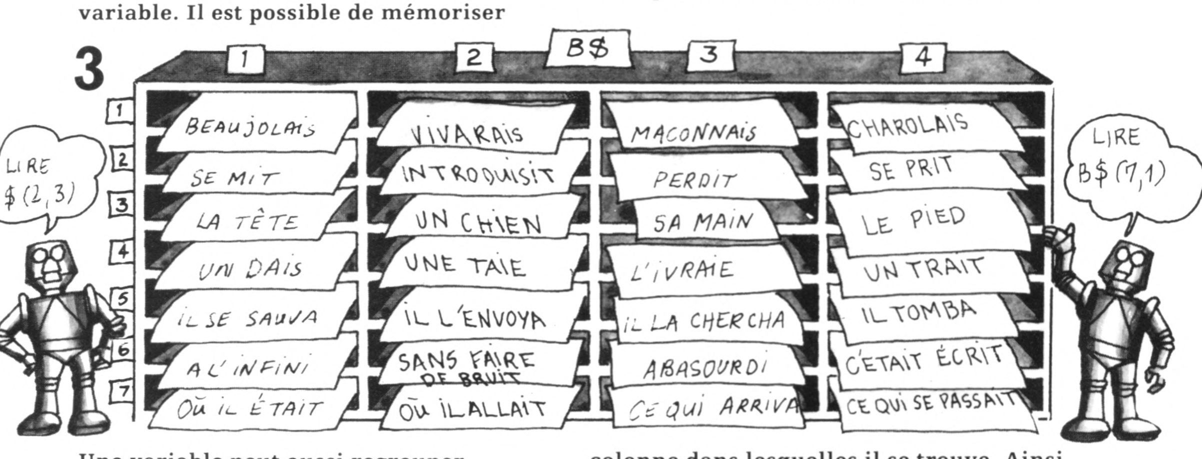
2 Entrer les données

```
50 READ A$

180 DATA C'ÉTAIT UN JEUNE HOMME DU,
    QUI, DANS
190 DATA UN SOIR À LA NUIT,
    ET IL NE SUT JAMAIS
```

Pour entrer les données, on peut utiliser les instructions **READ** et **DATA**. Chaque fois que l'ordinateur rencontre l'instruction READ, il prend un nouvel article à la ligne DATA et le mémorise en variable. Il est possible de mémoriser tous les articles en une même variable A$. Toute variable contenant plus d'un article s'appelle un tableau et chaque article est repéré par un numéro, par exemple READ A$(3) donne « dans »*.

3



Une variable peut aussi regrouper plusieurs rangées de données. C'est alors un tableau à deux dimensions. Vous pouvez ainsi mémoriser tous les mots du poème. Repérez chaque article par le numéro de la rangée et le numéro de la colonne dans lesquelles il se trouve. Ainsi READ B$(4,2) donne « une taie » et READ B$(6,3) « abasourdi ».
En utilisant une variable numérique, vous pouvez aussi stocker des nombres dans un tableau.

4 Mettons les données en variables

```
10 FOR I=1 TO 7 }- I renvoie au n° de la rangée
20 FOR J=1 TO 4 }- J renvoie au n° de la colonne
30 READ B$(I,J)
40 NEXT J
50 NEXT I
60 DATA BEAUJOLAIS, VIVARAIS,
   MACONNAIS, CHAROLAIS
70 DATA SE MIT, INTRODUISIT, PERDIT, SE PRIT
80 DATA LA TÊTE, UN CHIEN, SA MAIN, LE PIED
90 DATA UN DAIS, UNE TAIE, L'IVRAIE, UN TRAIT
100 DATA IL SE SAUVA, IL L'ENVOYA, IL LA CHERCHA, IL TOMBA
110 DATA À L'INFINI, SANS FAIRE DE BRUIT, ABASOURDI,
    C'ÉTAIT ÉCRIT
120 DATA OÙ IL ÉTAIT, OÙ IL ALLAIT, CE QUI ARRIVAIT,
    CE QUI SE PASSAIT
```

Pour que chaque article passe en variable, il faut modifier les nombres entre parenthèses après READ. Des boucles peuvent le faire : on utilise des boucles imbriquées pour B$, la boucle J (numéro de colonne) se répétant 4 fois pour chaque passage de la boucle I (numéro de rangée).



*Ce programme ne fonctionnera pas sur un SINCLAIR qui traite les variables de façon différente. Vous en saurez plus à la page suivante.

5 Créons des espaces pour les variables

```
5 DIM K$(5)
10 FOR I=1 TO 5
20 READ K$(I)
30 NEXT I
40 STOP

60 DATA CHIEN, CHAT, PUCE, MOUCHE
```

Dimension de la variable. Exemple : 5 articles dans une rangée. (5 DIM K$(5))

Cette ligne inscrit la donnée en K$ à chaque passage de la boucle. (20 READ K$(I))

Ce programme comporte un bug. Il n'y a pas assez d'articles pour la dimension de la variable.

En début de programme, on doit indiquer à l'ordinateur quelle sera la dimension de la variable. Pour cela, utilisez le mot DIM suivi du nom de la variable et du nombre d'articles qu'elle comporte. Exemple : DIM K$(5). Dans un tableau à deux dimensions, on indique à l'ordinateur le nombre de rangées et le nombre de colonnes de la variable : DIM C$(5,3). Il faut toujours que le nombre d'articles et le nombre indiqué coïncident, ou c'est un bug.

6 Affichons les données

```
200  LET A=0
210  LET B=0
220  LET A=A+1
230  IF A=6 THEN STOP
240  PRINT A$(A)
250  LET B=B+1
260  LET N=INT(RND(1)*4+1)
270  PRINT B$ (B,N)
280  IF B=2 THEN GO TO 250
290  IF B=5 THEN GO TO 250
300  GO TO 220
310  END
```

A compte le nombre de fois où cette partie du programme est répétée. (220)

B compte les rangées de mots et vérifient qu'on utilise bien la bonne rangée pour chaque ligne de données. (250)

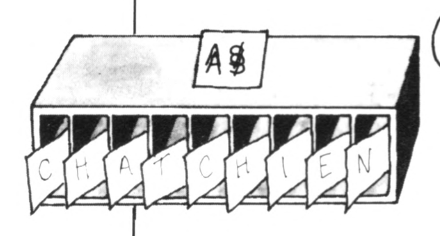
Les lignes 280 et 290 font afficher à l'ordinateur les mots sélectionnés dans une rangée avant de visualiser la ligne de données suivante.

Donne l'ordre à l'ordinateur d'aller afficher la ligne de données suivante. (300)

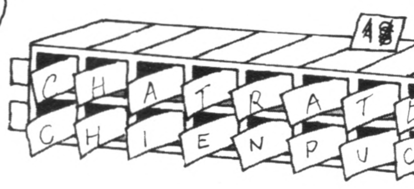
Ce programme est nécessaire pour que l'ordinateur affiche mots et données dans le bon ordre. Cette partie du programme se répète 5 fois. À chaque passage, l'ordinateur affiche la ligne de données A et un mot de la rangée B. La sélection des mots est faite par un nombre aléatoire N.

L'ordinateur SINCLAIR et les variables

Ce programme ne fonctionne pas tel quel sur les ordinateurs SINCLAIR, car ils traitent les chaînes différemment.







Pour indiquer à un SINCLAIR de sélectionner une donnée en particulier, il faut lui fournir la position du premier et du dernier caractère du mot retenu. C'est en somme la même méthode que pour LEFT$ et RIGHT$ page 32. Pour les tableaux à deux dimensions, il faut donner à l'ordinateur le nombre de rangées et le nombre de caractères. Par exemple A$(2,6 TO 9) c'est « PUCE ».

En début de programme, vous devez indiquer à l'ordinateur le nombre de rangées et le nombre de caractères par rangée. DIM A$(2,9) explique qu'il y a deux rangées de neuf caractères chacune. Toutes les rangées du tableau doivent comporter le même nombre de caractères.

Programme complet de poésie

On peut maintenant rassembler tous ces éléments pour reconstituer enfin le programme de poésie in extenso. La première partie enregistre les données (lignes 10 à 90), la seconde affiche le poème (lignes 200 à 310). À chaque fois que vous lancez le programme, vous obtenez une nouvelle version du poème, puisque le nombre aléatoire N fait sélectionner des mots différents à l'ordinateur.

```
10 DIM A$(5)
20 DIM B$ (7,4)
30 FOR I=1 TO 7
40 FOR J=1 TO 4
50 READ B$(I,J)
60 NEXT J
70 NEXT I
80 DATA BEAUJOLAIS, VIVARAIS, MACONNAIS, CHAROLAIS
90 DATA SE MIT, INTRODUISIT, PERDIT, SE PRIT
100 DATA LA TÊTE, UN CHIEN, SA MAIN, LE PIED
110 DATA UN DAIS, UNE TAIE, L'IVRAIE, UN TRAIT
120 DATA IL SE SAUVA, IL L'ENVOYA, IL LA CHERCHA, IL TOMBA
130 DATA À L'INFINI, SANS FAIRE DE BRUIT, ABASOURDI, C'ÉTAIT ÉCRIT
140 DATA OÙ IL ÉTAIT, OÙ IL ALLAIT, CE QUI ARRIVAIT, CE QUI SE PASSAIT
150 FOR I=1 TO 5
160 READ A$(I)
170 NEXT I
180 DATA C'ÉTAIT UN JEUNE HOMME DU, QUI, DANS
190 DATA UN SOIR À LA NUIT, IL NE SUT JAMAIS
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN 310
240 PRINT A$(A)
250 LET B=B+1
260 LET N=(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=2 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

Les lignes 10 et 20 réservent l'espace pour les variables.

Boucles imbriquées pour mettre les données en B$.

Les lignes 80 à 140 contiennent tous les mots à entrer en B$.

Boucle servant à mettre les données en A$.

Les lignes 180 et 190 contiennent toutes les données à entrer en A$.

Cette instruction fait s'afficher la donnée mémorisée en A$, rangée A. (ligne 240)

Cette instruction fait s'afficher le mot mémorisé en B$, rangée B, colonne N. (ligne 270)

Le programme s'arrête à la ligne 230 quand A=6. Mais certains ordinateurs réclament tout de même un END.

Exemples

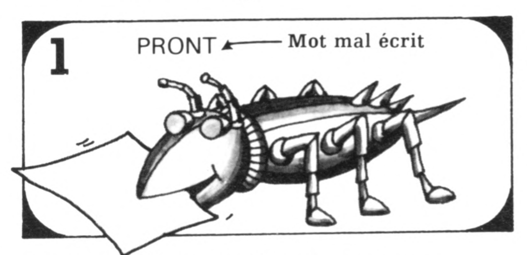
C'ÉTAIT UN JEUNE HOMME DU	C'ÉTAIT UN JEUNE HOMME DU
VIVARAIS	CHAROLAIS
QUI	QUI
SE MIT	PERDIT
LA TÊTE	LE PIED
DANS	DANS
UN DAIS	UNE TAIE
UN SOIR À LA NUIT	UN SOIR À LA NUIT
IL SE SAUVA	IL TOMBA
SANS FAIRE DE BRUIT	C'ÉTAIT ÉCRIT
ET IL NE SUT JAMAIS	ET IL NE SUT JAMAIS
CE QUI ARRIVAIT	OÙ IL ALLAIT

Voilà deux des 16 384 versions possibles de la poésie ! Si, en lançant le programme, vous obtenez toujours les mêmes données, regardez dans votre manuel ce que vous devez faire pour que votre ordinateur tire des nombres différents. Sur certains modèles, les nombres aléatoires sont les mêmes à chaque fois qu'on met en marche l'ordinateur. En utilisant la fonction GOTO, par exemple, vous éliminerez cet inconvénient.

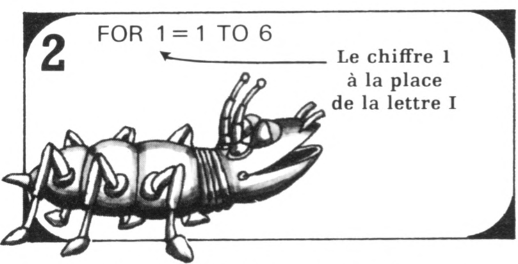
Quelques trucs de programmation

Dans ces deux pages, vous trouverez quelques trucs pour vous aider à écrire vos propres programmes, ainsi qu'une liste des bugs les plus courants. On vous explique le pourquoi des bugs en partant des plus évidents jusqu'aux plus retors.

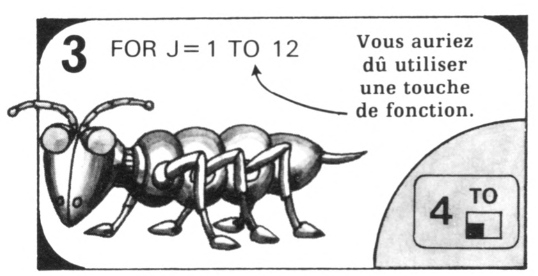
Trouver le bug



Cherchez d'abord les fautes de frappe dans les instructions de BASIC. Si un des termes est mal écrit, l'ordinateur ne le reconnaîtra pas.



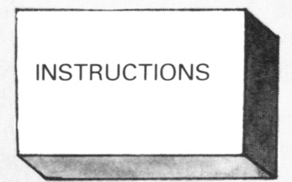
Vérifiez les O et les 0, les 1 et les I pour être certain qu'il n'y a pas eu de confusion.



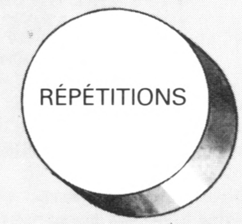
Si vous avez un SINCLAIR, vérifiez que vous n'avez pas tapé lettre à lettre une instruction qui s'effectue en appuyant sur une touche spéciale.

Programmer

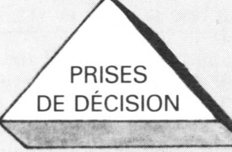
Quand on écrit des programmes, il faut se rappeler qu'un ordinateur accomplit trois tâches principales : instructions simples, répétitions et prises de décision. Ce sont les pierres angulaires des programmes.



LET A=3
LET N=N+1
PRINT A/T
PLOT (X,Y)

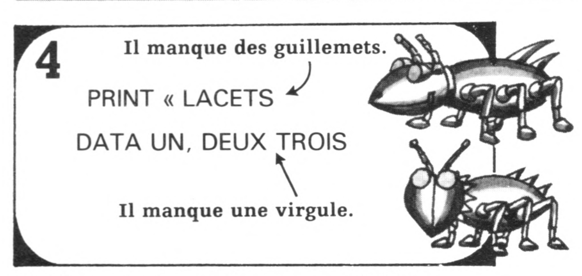


FOR J=1 TO 6
20 LET A=1
30 IFA<10 THEN GOTO 100



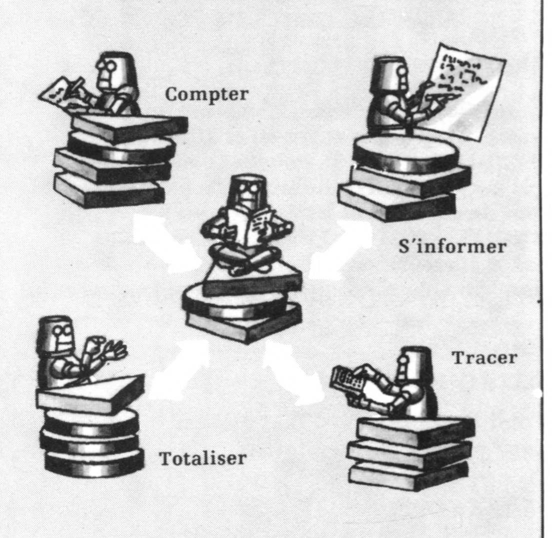
IF X=Y THEN STOP
IF K$=« BONJOUR » THEN PRINT A

Ce livre vous a expliqué toutes les instructions essentielles du BASIC, qui permettent d'exécuter ces fonctions. Lorsque vous écrivez un programme, déterminez ce dont l'ordinateur a besoin à chaque étape, puis appliquez les instructions correspondantes.

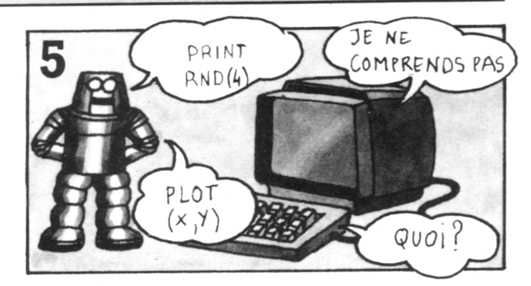


Vérifiez que vous n'avez pas oublié guillemets, virgules ou autres. Contrôlez soigneusement les lignes de programme très chargées en signes.

On peut écrire un même programme de plusieurs façons; certaines sont plus claires et rapides que d'autres. Quand vous devez établir un long programme, il est bon de le diviser en plusieurs parties et d'utiliser des sous-programmes pour chaque type de fonction. Le corps central peut parfois se résumer à quelques instructions, décisions et répétitions qui commandent les sous-programmes.



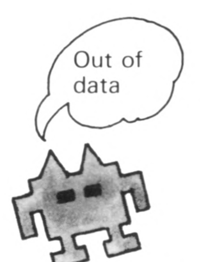
Morceler ainsi un programme le rend beaucoup plus facile à tester. On peut rechercher les erreurs, portion par portion, sans avoir à lancer le programme en entier. N'oubliez pas d'étiqueter chaque partie avec une ligne REM, – pour vous y retrouver.



Vérifiez que vous utilisez la bonne instruction pour RND, PLOT, et CLS. N'oubliez pas de passer en mode graphique.

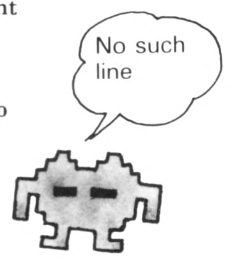
Message d'erreurs

Tous les ordinateurs affichent un message lorsqu'il y a un bug. Ces messages sont expliqués dans votre notice. Voici les messages les plus courants :



◀ Il n'y a pas assez de rubriques en DATA. Peut-être avez-vous oublié une virgule entre deux articles que l'ordinateur a alors pris pour un seul.

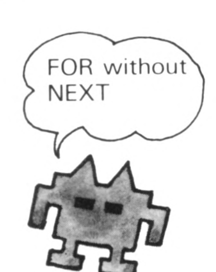
▶ GO TO ou GOSUB envoient l'ordinateur à une ligne qui n'existe pas. Vous avez effacé cette ligne en donnant le même numéro à une autre ligne ou, tout simplement, vous vous êtes trompé en tapant le nombre.



No such variable

◀ Vous rencontrerez ce message sur le BBC ou le SINCLAIR. Il indique généralement que vous n'avez pas initialisé une variable avant de l'utiliser avec une ligne comme LET C=O ou Let C=«   ».

▶ Dans votre boucle, la ligne NEXT manque. Ou vous vous êtes trompé dans le nom de la variable, ou vous avez écrit le chiffre 1 à la place de la lettre I.



Un dernier mot

Certains bugs sont très difficiles à détecter. Pourtant, si le programme ne tourne pas, c'est qu'il y a une erreur quelque part... Si vous ne la trouvez pas, tapez à nouveau les lignes particulièrement délicates ou compliquées. Peut-être que le programme se mettra à fonctionner sans que vous puissiez vraiment déterminer d'où provenait l'erreur.

Réponses aux casse-tête

Page 15
Un message à votre nom

```
10 PRINT « QUEL EST TON NOM »
20 INPUT N$
30 PRINT « BONJOUR »
40 PRINT N$
50 PRINT « COMMENT VAS-TU »
```

Page 17
1. Programme d'addition

```
10 LET A=9
20 LET B=7
30 PRINT A*B
40 PRINT A/B
50 LET A=A+1
60 LET B=B+3
70 PRINT A*B, A/B
80 END
```

La virgule laisse un espace.

2. Table de multiplication

```
30 PRINT A; « FOIS »; B; « FONT »; A*B
40 PRINT A; « DIVISÉ PAR »;
        B; « ÉGALE »; A/B
```

Espaces

3. Sur une même ligne

```
10 PRINT « QUEL EST TON NOM »
20 INPUT N$
30 PRINT « BONJOUR »; N$;
        « COMMENT VAS-TU »
```

Page 18
Opérations

```
10 PRINT « COMBIEN FONT 7 FOIS 7 »
20 INPUT A
30 IF A=49 THEN PRINT « JUSTE »
40 IF A<>49 THEN PRINT « NON »; 7*7
```

N'oubliez pas les points-virgules.

Page 19
Quel est mon âge?

Il faut transformer la ligne 30 et ajouter la ligne 35 :

```
30 IF G<14 THEN PRINT
« PLUS VIEUX QUE ÇA »
35 IF G>14 THEN PRINT
« PLUS JEUNE QUE ÇA »
```

Page 23
Contrôleur de boucle

```
5 LET C=O
45 LET C=C+1
50 IF C<6 THEN GOTO 10
```


L'initiale

Voici un exemple pour la lettre L.

```
10 LET X=15
20 LET Y=30
30 PLOT (X,Y)
40 LET Y=Y-1
50 IF Y>5 THEN GOTO 30
60 LET X=X+1
70 PLOT (X,Y)
80 IF X<45 THEN GOTO 60
90 END
```

Page 24
Nombre aléatoire

La formule permettant d'obtenir un nombre aléatoire entre 10 et 20 est : INT(RND(1)*11+9). Pour les ordinateurs qui acceptent un nombre entre parenthèses tout de suite après RND, cela donne RND(11)+9. Il y a 11 nombres possibles entre 10 et 20; on doit donc tirer un nombre aléatoire entre 1 et 11, puis ajouter 9.

Page 25
Attaque spatiale

Voici les lignes qu'il faut ajouter pour obtenir ce résultat :

```
15 LET S=O
75 IF X=A*B THEN LET S=S+1
95 PRINT « VOUS AVEZ ABATTU »
    ; S; « EXTRA-TERRESTRES »
```

Page 27
1. Table de 8

```
10 PRINT « LA TABLE DE MULTIPLICATION
   PAR HUIT »
20 FOR J=1 TO 12
30 PRINT J; « x8= »; J*8
40 NEXT J
```

Page 27
2. Table de N

```
10 INPUT « TAPER UN NOMBRE »; N
20 PRINT « VOICI LA TABLE
   DE MULTIPLICATION PAR »; N
30 FOR I=1 TO 12
40 PRINT I; « FOIS »; N; « FONT »; I*N
50 NEXT I
60 INPUT « UN AUTRE NOMBRE
   (O OU N) »; M$
70 IF M$=« O » THEN GOTO 10
```

Sur le SINCLAIR ZX81, il faut séparer les PRINT et les INPUT.

Page 32
Le livre de l'ordinateur

LEFT$(A$,8): LE LIVRE
RIGHT$(A$,10): ORDINATEUR
MID$(A$,10,2): DE

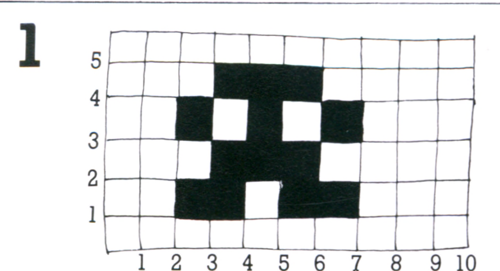
Page 34
Le nombre truqué

```
10 PRINT « PENSEZ À UN NOMBRE »
20 PRINT « MULTIPLIEZ-LE PAR 2, AJOUTEZ 4 »
30 PRINT « DIVISEZ PAR 2, AJOUTEZ 7 »
40 PRINT « MULTIPLIEZ PAR 8, ENLEVEZ 12 »
50 PRINT « DIVISEZ PAR 4, ENLEVEZ 11 »
60 PRINT « DITES-MOI COMBIEN IL RESTE »
70 INPUT N
80 PRINT « LE NOMBRE AUQUEL VOUS AVIEZ PENSÉ ÉTAIT »; (N-4)/2
```

Il ne faut pas oublier les parenthèses pour que l'ordinateur exécute d'abord la soustraction.

L'envahisseur extra-terrestre

1



Dessinez une forme simple d'envahisseur extra-terrestre sur une feuille de papier quadrillé.

2

4,5; 5,5; 6,5

3,4; 5,4; 7,4

4,3; 5,3; 6,3

3,2; 4,2; 6,2; 7,2

Puis relevez toutes les coordonnées des cases qui constituent la forme de l'envahisseur.

3

```
5 CLS
50 INPUT « DÉFINITION DE L'ÉCRAN, LARGEUR »; L
60 INPUT « HAUTEUR »; H
65 CLS
70 FOR I=0 TO H STEP H/6
80 FOR J=0 TO L STEP L/6

130 NEXT J
140 NEXT I
150 END
```

Si vous voulez que l'envahisseur apparaisse plus souvent à l'écran, remplacez 6 par un chiffre plus fort. (Si vous choisissez un nombre trop fort, c'est un bug.)

Insérez ici les lignes PLOT :

```
90 PLOT (J+3,I+2)
92 PLOT (J+4,I+2)
```

pour réaliser les deux cases en bas à gauche de la forme dessinée ici. Il faut une ligne de programme pour chaque case.

Recopiez le programme de répétition de dessins en remplaçant les lignes 10 à 40 par celles ci-dessus (vous n'avez pas besoin des lignes aléatoires qu'elles génèrent).
Il faut introduire vos instructions de tracé entre les lignes 80 et 140 (vous pouvez établir une nouvelle numérotation si besoin est). Pour chaque coordonnée, vous devez ajouter J au premier chiffre et I au second pour que le dessin se répète à l'écran.

Petit lexique du BASIC

Voici une liste des expressions de BASIC utilisées dans ce livre. Vous trouverez une rapide explication de leur signification. Certains d'entre eux, comme CLS, ne correspondent pas à tous les ordinateurs : vous les reconnaîtrez à une petite étoile placée devant eux. Si vous avez un ordinateur, vérifiez ces instructions dans votre notice.

*** BREAK : Sur certains ordinateurs, cette instruction arrête le programme. Mais attention, sur d'autres machines elle efface complètement le programme de la mémoire de l'ordinateur. Vous devez donc utiliser à la place ESCAPE, STOP, ou le mot indiqué dans votre notice.**

*** CLS : Vide l'écran.**

*** DATA : Série d'articles comme des mots ou des nombres, qui sont mémorisés en variables. Voir READ.**

DIM : Indique à l'ordinateur quel espace il doit réserver à une variable. Exemple DIM A$(5,4) indique que la variable occupera cinq rangées de quatre colonnes.

*** EDIT : Permet de modifier une ligne dans un programme, sans avoir à la retaper en entier.**

*** END : Indique la fin du programme à l'ordinateur. Certains ont toujours besoin de cette instruction; d'autres, comme le BBC ou le SINCLAIR, peuvent s'en passer.**

FOR... NEXT : Fait se répéter une boucle dans un programme autant de fois qu'il est indiqué.

GOSUB : Fait quitter à l'ordinateur le programme principal pour rejoindre un sous-programme (en anglais : subroutine) afin d'effectuer une tâche particulière.

GOTO : Envoie l'ordinateur à une autre ligne du programme.

IF... THEN : Compare des données (par exemple nombres, mots ou contenu de variables) et agit en fonction des résultats.

INPUT : Fait demander par l'ordinateur une donnée en cours de programme.

INT : Transforme un nombre décimal en un nombre entier en laissant tomber tout ce qui se situe à droite de la virgule. Exemple : INT(3,40)=3

*** LEFT$: Indique à l'ordinateur de prendre un certain nombre de caractères situés à l'extrémité gauche d'une chaîne alphanumérique. LEFT$(A$,4) indique de prendre les 4 premiers caractères à gauche de la chaîne.**

LEN : Indique la longueur d'une chaîne, par exemple le nombre de caractères dans une variable.

Petit lexique de l'ordinateur

Bug : Erreur dans le programme.

Chaîne : Série de caractères à mémoriser en variable comme « saucisse » ou « ABC123 ».

Curseur : Point lumineux, parfois clignotant, qui apparaît à l'écran pour indiquer où vâ s'afficher le caractère suivant.

Erreur de syntaxe : Erreur de BASIC dans le programme.

Graphique : Mode de visualisation à l'écran.

Kilo octets : Unité de mesure des mémoires d'ordinateur. Un KO contient 1 024 octets : Dans la plupart des machines, un caractère occupe un octet.

Organigramme : Plan du programme qui en situe les étapes principales. C'est souvent une aide à la programmation.

PIXEL : Point lumineux. Petits carrés que l'ordinateur allume sur l'écran pour tracer des dessins.

LET : Étiquette une variable en mémoire et y place une information. Exemple : LET N=4 ou LET B$="CHAT".

* LIST : Affiche la liste du programme à l'écran.

* MID$: Indique à l'ordinateur de prendre un certain nombre de caractères situés au milieu d'une chaîne alphanumérique. Exemple : MID$(A$,4,3) indique de prendre trois lettres à compter de la quatrième lettre de A$.

NEW : Vide la mémoire de l'ordinateur pour pouvoir y entrer de nouvelles données.

* NEWLINE : Indique à l'ordinateur que l'on a fini d'entrer une ligne d'instructions ou une donnée. Sur d'autres ordinateurs, la touche s'appelle RETURN ou ENTER.

NEXT : Voir FOR... NEXT.

* PLOT : Indique à l'ordinateur d'allumer un point lumineux : PLOT(X,Y) signifie que l'ordinateur doit éclairer à l'écran le point dont les coordonnées sont X et Y.

PRINT : Indique à l'ordinateur d'afficher un message à l'écran.

* READ : L'ordinateur doit aller chercher une donnée en DATA, puis la mémoriser en une variable. Voir DATA.

* READY : Message délivré par certains ordinateurs lorsqu'ils sont prêts à recevoir une nouvelle instruction.

REM : L'ordinateur ne tient pas compte des lignes commençant par REM, mais les affiche dans la liste. Elles permettent de se repérer dans le programme.

RETURN : Se place à la fin d'un sous-programme. Renvoie l'ordinateur à l'instruction qui suit celle où il a quitté le programme principal. Voir GOSUB.

* RIGHT$: Indique à l'ordinateur de prendre un certain nombre de caractères à l'extrémité droite d'une chaîne alphanumérique. RIGHT$(A$,4) indique de prendre les quatre derniers caractères à droite de la chaîne.

* RND : Tire un nombre aléatoire.

RUN : Lance le programme.

SQR : Demande à l'ordinateur de calculer la racine carrée d'un nombre.

STEP : S'utilise avec FOR ... NEXT dans les boucles. Indique à l'ordinateur quand répéter la boucle.

STOP : S'utilise à l'intérieur d'un programme pour en arrêter le déroulement.

THEN : Voir IF ... THEN.

* UNPLOT : Indique à l'ordinateur d'éteindre un point lumineux.

Programme : Liste numérotée d'instructions, qui indique à l'ordinateur comment exécuter une tâche.

Prompt : Point d'interrogation qui apparaît lorsque l'ordinateur demande des instructions à la suite d'un INPUT.

RAM : (Random Access Memory) Mémoire de l'ordinateur où sont stockés programmes et données. Tout ce qui est mémorisé en RAM s'efface quand on coupe l'ordinateur.

ROM : (Read Only Memory) Mémoire permanente qui contient les instructions de fonctionnement de l'ordinateur.

Sous-programme : Partie d'un programme à laquelle est assignée une tâche. On l'utilise plusieurs fois dans un même programme.

Tableau : Ensemble de variables contenant plusieurs données.

Unité centrale : Unité de traitement de l'ordinateur qui accomplit le travail et contrôle toutes les opérations.

Variable : Espace-mémoire étiqueté qui contient une information.

Pour aller plus loin

La meilleure façon d'apprendre à programmer, c'est d'entrer vos programmes sur un ordinateur. Si vous n'en avez pas, peut-être pouvez-vous en trouver un à utiliser quelque part. Demandez autour de vous, dans votre établissement, ou rejoignez un club d'utilisateurs et voyez si vous pouvez emprunter du temps-machine.

On peut beaucoup apprendre sur la programmation en lisant et en entrant sur un clavier des programmes écrits par d'autres.

Index

Loi n° 49.956 du 16 juillet 1949
sur les publications destinées à la jeunesse - dépôt 10-85
Dépôt légal n° 1389-10-1985
ISBN: 2.01.009363.1 29.62.0330.04
Imprimé en Belgique par Proost, Turnhout, Belgique.

GUIDES PRATIQUES DE MICRO-INFORMATIQUE

Que ne peut-on pas faire avec un ordinateur ! Calculer, bien sûr, mais aussi lui poser des questions, écrire des poèmes, jouer à quantité de jeux plus palpitants les uns que les autres, composer même de la musique...
Petits guides pratiques d'introduction à la micro-informatique, les ouvrages de cette nouvelle collection font découvrir toutes les possibilités qu'offrent les micro-ordinateurs. Ils initient au langage et au fonctionnement de l'ordinateur, apprennent à programmer et – pourquoi pas ? – à créer des programmes originaux ! La clarté du texte, la gaieté des couleurs, la drôlerie des dessins, tout est conçu dans ces livres pour faire de cette initiation un plaisir.








Dans la même collection







29.20 FF TTC


9 782010 093630

Imprimé en Belgique

29/0330/0
85/X



Document numérisé avec amour par :

AMSTRAD CPC

MÉMOIRE ÉCRITE

ACME

https://acpc.me/